AFIT/GE-93D-05

AD-A274 082

DTIC
S ELECTE D
DEC 23 1993
E

SATELLITE ATTITUDE DETERMINATION USING LINEAR COMBINATION OF MODELS

THESIS
Martin Stuart Chin
Captain, USAF

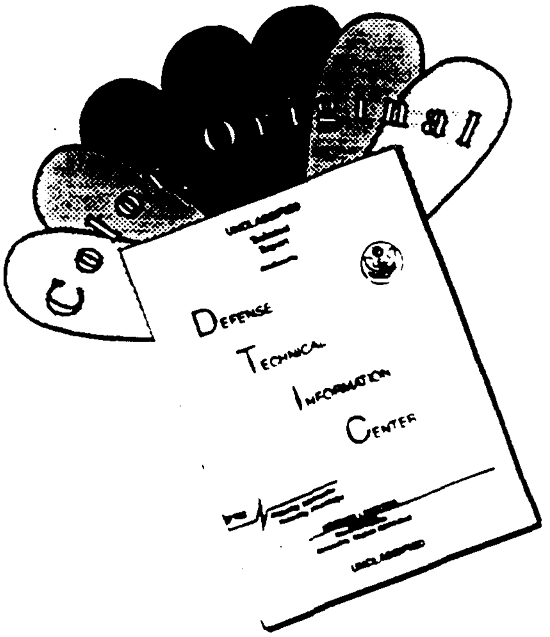
AFIT/GE-93D-05

93-31013

Approved for public release; distribution unlimited

93 12 22 126

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF COLOR PAGES WHICH DO NOT REPRODUCE LEGIBLY ON BLACK AND WHITE MICROFICHE.

AFIT/GE-93D-05

SATELLITE ATTITUDE DETERMINATION USING LINEAR COMBINATION OF MODELS

THESIS

Presented to the Faculty of the Graduate School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Martin Stuart Chin, B.S.E.E., M.S.S.M.
Captain, USAF

December 1993

Approved for public release; distribution unlimited

DTIC
COPY
INSPECTED
6

Accesion For
NTIS CRA&I ☒
DTIC TAB ☐
Unannounced ☐
Justification ____

By ____
Distribution /

Availability Codes

Dist	Avail and / or Special
A-1	

Preface

I must have sinned terribly in a previous life to deserve a tour at AFIT. It is with much rejoicing that I write this preface, and approach the end of this thesis effort. That being said, I must also say that my time at AFIT has been a most edifying experience. In many ways, this thesis has taught me much more than engineering.

The realization of this thesis would not have been possible without the guidance of Captain Dennis Ruck, my advisor, and Drs. Steven Rogers and Mark Oxley, the members of my thesis committee. Thank you.

I would also like to recognize the "back corner crew": Bob MacDonald, Gary Shartle, Neale Prescott, Curtis Martin, John Keller, and Kim McCrae. You made the lab much less dreary. Thanks also to Gershwin, Helen, and Tonya. I love you very much.

Martin Stuart Chin

Table of Contents

Page

Preface ii

List of Figures vi

List of Tables viii

Abstract ix

I. Introduction 1
  - 1.1 Background 1
  - 1.2 Problem Statement 2
  - 1.3 Scope 2
  - 1.4 Methodology 3
  - 1.5 Overview of Thesis 5

II. Current Knowledge 6
  - 2.1 Introduction 6
  - 2.2 Transformation in Homogeneous Coordinates 6
  - 2.3 Projection in to 2D Images 10
  - 2.4 Linear Combination of Models 11
  - 2.5 Previous Applications 15
    - 2.5.1 Recognition 15
    - 2.5.2 Image Registration 16

III. Linear Combination for Attitude Determination 17
  - 3.1 Introduction 17
  - 3.2 Linear Combination using 4D Homogeneous Coordinates 17

Page

3.2.1 Matrix Representation of Images 17
3.2.2 Basis Matrices 18
3.2.3 Choosing Basis Images 19
3.3 Determining Alignment Parameters From the Coefficients 20
3.4 Modeling in *Mathematica* 26
3.5 Implementation on Silicon Graphics Workstation 28
3.5.1 3D Model 29
3.5.2 Model Data Manipulation 29
3.5.3 Parameter Extraction 32

IV. Performance Analysis 37
4.1 Tolerance 37
4.2 Alignment Point Error 40
4.2.1 Fixed Magnitude Error 40
4.2.2 Statistical Position Error 43
4.3 Conclusion 54

V. Conclusions and Recommendations 56
5.1 Research Conclusions 56
5.2 Contributions 57
5.3 Recommendations 58
5.3.1 Alternative Parameter Extraction Techniques 58
5.3.2 Potential Improvements 59
5.4 Summary 60

Appendix A. User's Guide 61
A.1 Model Manipulation 61
A.1.1 Input File Formats 61
A.1.2 Execution 65

Page
A.2 Alignment Parameter Extraction 67
A.2.1 Input File Formats 67
A.2.2 Execution 68
A.3 Performance Analysis 71
A.3.1 Output File Format 71
A.3.2 Execution 71

Appendix B. *Mathematica* Source Code 73

Appendix C. Software Reference 78
C.1 Makefile 78
C.2 Parameter Extraction: *matcher.c* 79
C.3 Supporting Subroutines: *mysubs.c* 80
C.4 Numerical Routines: *calcs.c* 80
C.5 NSM Reader: *rdfile.c* 81
C.6 Image Generation: *writesat.c* 82
C.7 Rendering: *primitives.c* and *wprimitives.c* 82
C.8 Performance Analysis: *sens.c* 82
C.9 Summary 82

Bibliography 83

Vita 85

List of Figures

Figure Page

1. Simple object used in *Mathematica* modeling. 4
2. A translation transformation. 8
3. Perspective *vs.* orthographic projections. 11
4. Perspective and orthographic projections of *four*. 12
5. Basis images of *four*. 26
6. Random transformation of the object *four*. 27
7. Random and matched images. 29
8. Parameter extraction processing architecture 30
9. Image generator window on the SGI workstation. 31
10. Parameter extraction windows on the SGI workstation. 34
11. Logarithmic scatter plot of rms errors and convergence tolerance (inches). . . 39
12. Rotational parameters and convergence tolerance. 40
13. Two equivalent rotation modes. 41
14. Scale parameter errors and tolerance. 42
15. x translation parameter errors and tolerance. 42
16. y translation parameter errors and tolerance. 43
17. Introduced errors in alignment point. 44
18. Logarithmic scatter plot of rms errors and alignment point error (inches). . . 45
19. Rotational parameter error and alignment point error. 46
20. Scale parameter error and alignment point error. 46
21. Scatter plot of output errors for relative variance error case. (inches) 47
22. Rms error of all alignment points at output for relative variance error introduced at input of indicated alignment point. (rms inches vs. trial index) . . . 48
23. Histograms of rms output errors for all alignment points for relative variance error introduced at input of indicated alignment point. (rms inches) 49

Figure Page

24. Logarithmic scatter plot of output errors for absolute variance case. (inches) . 51
25. Rms error for all alignment points at output for absolute variance introduced at input of indicated alignment point . (inches vs. trial index) 52
26. Rms error histogram for all alignment points at output for absolute variance error introduced at input of indicated alignment point . (rms inches) 53
27. Numeric keypad functions in for pose manipulation. 66

List of Tables

Table Page
1. Data for performance analysis 37
2. Descriptive statistics for relative error case. 47
3. Descriptive statistics for absolute variance case. 50

Abstract

In satellite analysis, it is often necessary to determine the attitude of a satellite from a two dimensional image of the satellite. This thesis describes work performed to determine the attitude of a satellite relative to a reference position. The attitude is described in terms of the scaling, rotation and translation of the reference that will result in a pose that can be projected to form the image of interest. The techniques used are based on finding coefficients for the linear combination of basis images and decomposition of a composite transformation matrix. The methods were initially modeled in *Mathematica* and applied to simple objects, then implemented on a Silicon Graphics Workstation and applied to a model of an actual satellite. An accurate estimate of the pose of a satellite in a 2D image can be estimated based on the information contained in the image and a 3D model of the satellite or a set of images of the satellite at known orientations. Results of a performance analysis of the SGI implementation, a user's guide, source code in *Mathematica* and a software reference are included.

SATELLITE ATTITUDE DETERMINATION USING LINEAR COMBINATION OF MODELS

I. Introduction

1.1 Background

Mankind's exploitation of space through the use of satellites has brought many benefits. From their high altitude orbits, they observe the earth and relay data to their masters. Earth, unfortunately, is a planet of many masters. Each of these masters has a keen interest in observing the activities of not only their own satellites, but those belonging to others as well.

An important aspect in the observation of satellites is the determination of attitude. A satellite's attitude or orientation in space provides many clues to its mission. From attitude information, the aiming of antennas, sensors, or thrusters may be determined. Then, combining attitude and location data, one can make an educated guess at what it is looking, and/or to whom it is talking.

Satellite observations can include imagery collected from the ground. In some cases, this may be the only data available. An analyst is then faced with estimating the orientation of a three-dimensional object in three-dimensional space from a two-dimensional image. While rough estimates are possible (for example, "it is pointing down"), more precise estimates ("it is pointing down at 63°") are difficult. A tool for making precise estimates could be very useful.

This thesis will show that it is possible to make a precise estimate of the orientation of a known object, in this case a satellite, shown in a two-dimensional image.

1.2 Problem Statement

This research focuses on creating a computerized aid for precisely estimating the attitude of a satellite represented in a two-dimensional image. The tool requires a base of information about the object of interest. In this case, the information is in the form of a 3D geometric model of the object. The tool can manipulate the model to create views of the object at arbitrary orientations, forming basis images and simulated sensor images from which recognition and attitude determination can be performed. The basis images and information about the attitude of the object in them provide sufficient information for recognition and determination of attitude in the simulated sensor image.

1.3 Scope

This research is based on Ullman and Basri's work in depicting and recognizing objects as linear combinations of models (13). Although this thesis is closely related to and discusses recognition, it is important to note that the focus of this research is not the recognition of objects. Rather, this work addresses the steps subsequent to recognition in which the orientation of the object is determined.

A basic assumption throughout this thesis is that the identity of the object is known. In many applications, this assumption is not unreasonable. In this particular application, the identity of the satellite can be determined from intelligence and ephemeris data on specific satellites. This being the case, the recognition portions of the algorithm were tested under relatively benign conditions. No attempt was made to fool the system by presenting an unexpected object or distorted views.

Additional assumptions include the structure of the object. Here it is assumed that the object has at least four distinct points or vertices that can be seen at most orientations. These vertices are selected beforehand and used as alignment points. Since the main criteria for selection of the alignment points is that they are in general position, their choice is almost arbitrary, and this assumption is valid for most objects. Additionally, the human operator is assumed to be able to estimate the position of hidden vertices.

The use of distinct vertices as alignment points may cause problems with objects where suitable vertices are not available. Such is the case with objects containing mostly curved surfaces, for instance, a Volkswagen Beetle. In their work, Ullman and Basri address the application of linear combination principles to objects with curved surfaces (13). The techniques described in this thesis can be extended to include objects of this type, but such an extension is deemed beyond the scope of this thesis.

The linear combination principles apply to 2D orthographic projections of 3D objects. Imaging sensors produce perspective projections. Orthographic projections produce images of an object without distortions due to size or distance, whereas perspective projections mimic the images formed by the human eye (3). In perspective projections, distant objects appear smaller than close objects of the same size. A more detailed discussion of the differences between orthographic and perspective projections is included in Chapter II. For the purposes of this thesis, it is assumed that orthographic projections closely approximate perspective projections. This assumption is valid given the scale of the objects being considered.

The location of the observer is also assumed to be known. Orientations will be determined in a frame of reference defined with respect to the observer and a 3D reference model. Transformations to other known frames of reference are then possible.

1.4 Methodology

This project was approached in two stages. First, the algorithms were modeled in *Mathematica* and tested on a simple object. Second, the process was implemented on a Silicon Graphics workstation and tested with a detailed model of a real satellite.

The first task of the modeling stage is the generation of a simple object that could be easily manipulated and transformed. This object, called *four* is shown in Figure 1. Next, it is verified that arbitrary 2D views the object can indeed be represented as linear combinations of 2D basis views, and that the coefficients used in the combination can be found analytically. The utility of the coefficients for identification is demonstrated, as described by Ullman and Basri. The method for determining the coefficients described by Ullman and Basri is modified

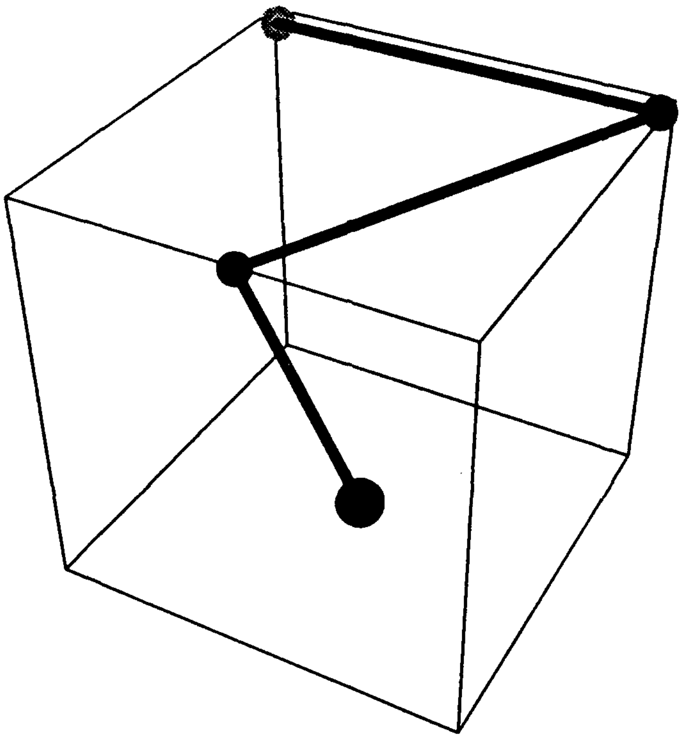


Figure 1. Simple object used in *Mathematica* modeling.

to work in four-dimensional homogeneous coordinates. 4D homogeneous coordinates were used so that all rigid transformations of the object (rotation about each axis, scaling, and translation) were considered. A method for determining the orientation of the objects from the coefficients is then devised. The method is tested to see if it can determine the attitude of the test object posed at random orientations. *Mathematica* was chosen because of its flexibility and support by many platforms. This portion of the project was run mainly on Apple Macintosh and NeXT computers. A complete printout of the *Mathematica* code is included as Appendix B.

Once the basic principles were tested, implementation on the Silicon Graphics workstation began. A constructive solid geometry model of the Defense Meterological Satellite (DMSP) satellite was obtained from the Philips Laboratory Satellite Assessment Center. New software to display and manipulate the model was necessary because of format of the model data and the requirements of the attitude determination software. The attitude determination software and graphical user interface were then built and tested with random poses of the satellite. A user's guide (Appendix A) and software reference (Appendix C) are included as appendices.

1.5 Overview of Thesis

This thesis is broken down into the following chapters:

Chapter II begins with a description of the representation and transformation of an object in 4D homogeneous coordinates. It then discusses the basics of Ullman and Basri's approach to linear combinations of models. Previous applications of the technique for recognition and image registration are briefly discussed.

Chapter III presents the specific techniques used in this application, including the generalization of the linear combination theory to encompass 4D homogeneous coordinates and all rigid transformations, determination of coefficients for the linear combination, and extraction of orientation information from the coefficients using Newton's method. Discussed in detail are the modeling performed in *Mathematica* and the SGI implementation.

Chapter IV provides the results of a qualitative performance analysis done on the SGI implementation of the attitude determination system.

Chapter V summarizes the results of the thesis and briefly discusses limitations of and new questions raised by the thesis, which may provide possibilities for future work on this topic.

The appendices include a software user's guide, and listings of the relevant *Mathematica* code, and a software technical guide.

II. Current Knowledge

2.1 Introduction

Determination of satellite attitude combines elements from computergraphics and artificial intelligence (AI) (3) (15). In order to achieve a useful blending of the two fields, an understanding of each is required.

The task embodies elements of AI in that it combines a broad range of information to solve a particular problem (15). Specifically, it combines information about the structure of an object, the way it looks in representative 2D views, and how rigid objects move and rotate in 3D space to estimate the object's attitude. The actual mechanisms by which the information is combined and manipulated are most often encountered in the field of computer graphics.

This chapter describes how objects are represented and manipulated in computer graphics. It then discusses Ullman and Basri's approach to image formation by the linear combination of basis images. Finally, it provides a brief overview of how the linear combination approach has been employed in the recognition of objects, and the use of transformation matrices for related applications.

2.2 Transformation in Homogeneous Coordinates

A three-dimensional object can be described by polygons that approximate its surfaces or lines that depict its edges. In either case, the description requires a series of points in three-dimensional space, where the points are the vertices of the polygons or the endpoints of the lines. Each point is described by a coordinate triple. Conventionally, the triple contains the $x-$, $y-$ and $z-$coordinates of the point in a Cartesian system, although cylindrical or spherical coordinates may be used.

Most transformations of concern here, specifically scaling and rotation, are linear transformations of a point in space. These transformations can be performed by writing the point as a position row vector, and multiplying by an appropriate 3 × 3 transformation matrix.

$$\hat{\boldsymbol{p}} = \boldsymbol{p}\boldsymbol{T}$$

where $\hat{\boldsymbol{p}}$ and $\boldsymbol{p}$ are vector triples of the new and old positions, respectively, and $\boldsymbol{T}$ is the transformation matrix.

Transformations that can be performed in this fashion include scaling and rotation about each axis using the following matrices:

$$\boldsymbol{S} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix}$$

$$\boldsymbol{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{bmatrix}$$

$$\boldsymbol{R}_y = \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{bmatrix}$$

$$\boldsymbol{R}_z = \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0 \\ -\sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where s_x, s_y, and s_z are the scale factors and θ_x, θ_y, and θ_z are the angles of rotation measured counterclockwise. Note that an object can be scaled by different amounts in each direction. This results in a warping effect and does not happen to non-deformable objects. For the purposes of this thesis, scaling will be restricted so that $s_x = s_y = s_z = s$. All transformations are performed with respect to the origin, and their matrix representations are always with respect to the standard Cartesian basis.

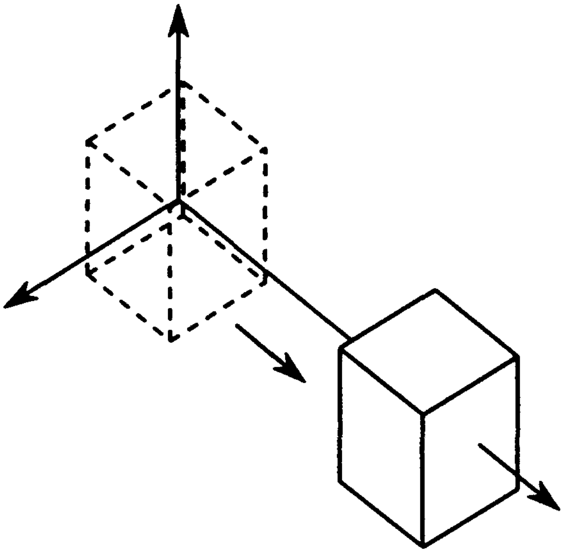


Figure 2. A translation transformation.

This set of transformations is, however, incomplete. Translations, or sliding movement without rotation or scaling (Figure 2) must be done by the addition of each translation amount to each coordinate.

$$\hat{p} = p + \triangle p$$

In order to allow transformations to be performed via matrix multiplication, a system of four-dimensional homogeneous coordinates is used (3). The 3D point is represented by a four element vector, (x, y, z, W). In this representation, (x, y, z, W) and (x', y', z', W') are the same point if and only if each element in one vector is a multiple of the corresponding element in the other. Thus the point $(2, 4, 5)$ can be represented by both $(2, 4, 5, 1)$ and $(4, 8, 10, 2)$.

When W is non-zero, the 4D point p can be homogenized to p_H by dividing through by W:

$$p_H = (\frac{x}{W}, \frac{y}{W}, \frac{z}{W}, 1).$$

Thus, a point is represented in homogeneous coordinates by simply adding a fourth coordinate of 1.

For 4D homogeneous coordinates, the transformation matrices are

$$\boldsymbol{S} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x & 0 \\ 0 & -\sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{R}_y = \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{R}_z = \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0 & 0 \\ -\sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

where s_x, s_y, and s_z and θ_x, θ_y, and θ_z are defined as above, and t_x, t_y, and t_z are the translations along the $x-$, $y-$, and $z-$ axes.

A complicated transformation can be performed by a series of n simple transformations of the types listed above.

$$\hat{\boldsymbol{p}} = \boldsymbol{p}\boldsymbol{T}_1\boldsymbol{T}_2\boldsymbol{T}_3\cdots\boldsymbol{T}_n$$

The matrices can be combined to form a composite transformation matrix, $\boldsymbol{T}_c$, such that

$$\hat{\boldsymbol{p}} = \boldsymbol{p}\boldsymbol{T}_c$$

where

$$\boldsymbol{T}_c = \boldsymbol{T}_1\boldsymbol{T}_2\boldsymbol{T}_3\cdots\boldsymbol{T}_n. \tag{1}$$

2.3 Projection in to 2D Images

The transformed points are then projected into a two-dimensional spaceto form an image. There are many ways to perform the projection. They fall basically into two categories: perspective and parallel (3). Perspective projections mimic the photographic systems and human vision. Objects at a distance appear smaller than an object of the same size that is closer, an effect called perspective foreshortening (3). Parallel projections are less realistic, but much easier to perform. Additionally, objects always appear to be the same size, regardless of distance, in parallel projections. The effect of distance can be simulated by introducing a scaling factor, to make distant objects appear smaller. In the special case of parallel projection that is employed here, the coordinates along the axes perpendicular to the projection plane is simply discarded. For projection into the xy plane, the z—coordinate and W are discarded. This is known as an **orthographic projection**. Single point perspective and orthographic projections of a familiar sight, railroad tracks, are compared in Figure 3. The perspective effect is exaggerated for clarity.

For objects and views of concern in this project, the orthographic and perspective views will be almost indistinguishable. Figure 4 shows the object *four* projected in perspective and orthographic. Besides the frames, which are not part of the image, the images are almost identical.

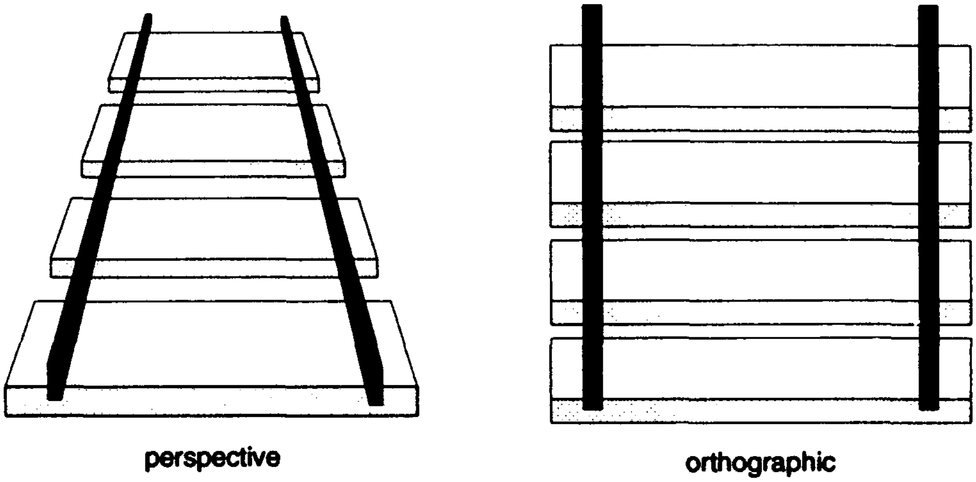


Figure 3. Perspective *vs.* orthographic projections.

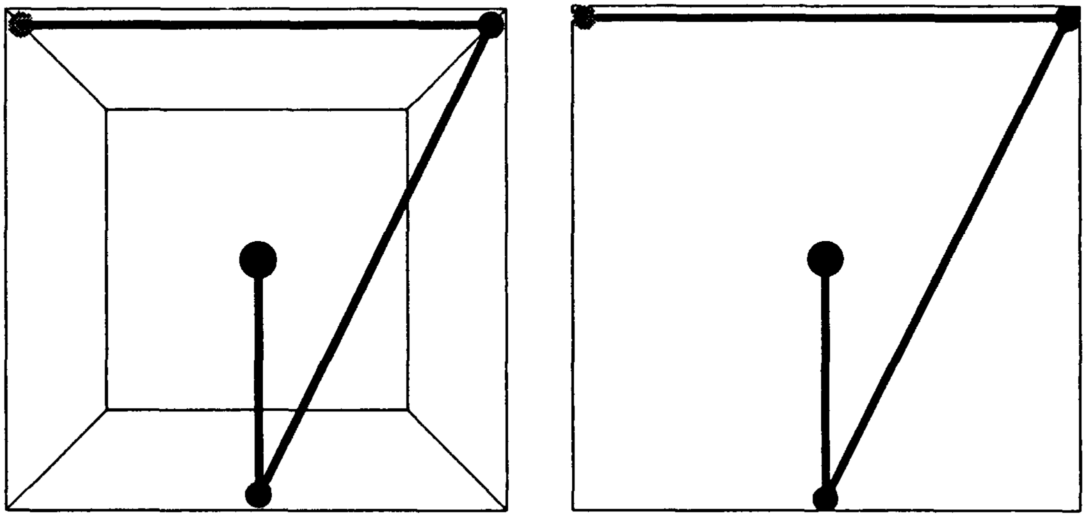
2.4 Linear Combination of Models

Given a 3D model and using the methods outlined above, one can readily create a 2D image of the object at any pose. It is less obvious, however, that the 2D image of the object at an arbitrary pose can be generated from a collection of 2D images of the same object at other poses. The collection of images from which a new image is made are called the basis images. Notionally, the new image is then formed from the linear combination of the basis images.

$$\hat{\boldsymbol{P}} = c_1\boldsymbol{P}_1 + c_2\boldsymbol{P}_2 + c_3\boldsymbol{P}_3 + \cdots + c_m\boldsymbol{P}_m$$

where $\hat{\boldsymbol{P}}$ is the new image and $\boldsymbol{P}_i$, $i = 1 \dots m$ are m basis images. The image, $\hat{\boldsymbol{P}}$, takes the form of an $n \times 2$ array whose rows are the x, y coordinate pairs of each of n points of interest on the object.

Ullman and Basri show that for the general case of rotation, translation and scaling in 3D space, an image of a rigid object can be expressed as a linear combination of four views of an object. Three of the four views must be independent, and the fourth may be derived from those three. The following is extracted from their proofs in *Recognition by Linear Combination of Models* (13):



perspective

orthographic

Figure 4. Perspective and orthographic projections of *four*.

Let $\boldsymbol{O}$ be a set of 3D points representing an object. Let $\boldsymbol{P}_1$, $\boldsymbol{P}_2$, and $\boldsymbol{P}_3$ be three images of the object generated by applying the 3×3 rotational (containing only rotations) transformation matrices $\boldsymbol{R}$, $\boldsymbol{S}$, and $\boldsymbol{T}$ to $\boldsymbol{O}$. Let $\hat{\boldsymbol{P}}$ be a fourth image of the same object obtained by applying the 3×3 transformation matrix $\boldsymbol{U}$ to $\boldsymbol{O}$. Let $\boldsymbol{r}_1$, $\boldsymbol{s}_1$, $\boldsymbol{t}_1$, and $\boldsymbol{u}_1$ be the first column vectors of $\boldsymbol{R}$, $\boldsymbol{S}$, $\boldsymbol{T}$, and $\boldsymbol{U}$, and let $\boldsymbol{r}_2$, $\boldsymbol{s}_2$, $\boldsymbol{t}_2$, and $\boldsymbol{u}_2$ be the second column vectors of $\boldsymbol{R}$, $\boldsymbol{S}$, $\boldsymbol{T}$, and $\boldsymbol{U}$. The positions of a point $\boldsymbol{p} = (x, y, z)$ such that $\boldsymbol{p} \in \boldsymbol{O}$ in the four images are given by:

$$\begin{array}{rcccl} \boldsymbol{p}_1 & = & (x_1, y_1) & = & (\boldsymbol{p}\boldsymbol{r}_1, \boldsymbol{p}\boldsymbol{r}_2) \\ \boldsymbol{p}_2 & = & (x_2, y_2) & = & (\boldsymbol{p}\boldsymbol{s}_1, \boldsymbol{p}\boldsymbol{s}_2) \\ \boldsymbol{p}_3 & = & (x_3, y_3) & = & (\boldsymbol{p}\boldsymbol{t}_1, \boldsymbol{p}\boldsymbol{t}_2) \\ \hat{\boldsymbol{p}} & = & (\hat{x}, \hat{y}) & = & (\boldsymbol{p}\boldsymbol{u}_1, \boldsymbol{p}\boldsymbol{u}_2) \end{array} \tag{2}$$

Claim: If both sets $\{\boldsymbol{r}_1, \boldsymbol{s}_1, \boldsymbol{t}_1\}$ and $\{\boldsymbol{r}_2, \boldsymbol{s}_2, \boldsymbol{t}_2\}$ are linearly independent, then there exist scalars a_1, a_2, a_3, b_1, b_2, and b_3 such that for every $\boldsymbol{p} \in \boldsymbol{O}$ it holds that:

$$\hat{x} = a_1 x_1 + a_2 x_2 + a_3 x_3$$
$$\hat{y} = b_1 y_1 + b_2 y_2 + b_3 y_3$$

Proof: $\{\boldsymbol{r}_1, \boldsymbol{s}_1, \boldsymbol{t}_1\}$ are linearly independent. Therefore, they span $\Re^3$, and there exist scalars a_1, a_2, and a_3 such that:

$$\boldsymbol{u}_1 = a_1 \boldsymbol{r}_1 + a_2 \boldsymbol{s}_1 + a_3 \boldsymbol{t}_1 \tag{3}$$

Transforming $\boldsymbol{p}$ by $\boldsymbol{U}$ gives $\hat{x}$:

$$\hat{x} = \boldsymbol{p}\boldsymbol{u}_1$$

Substituting (3) and multiplying by $\boldsymbol{p}$, it follows that:

$$\hat{x} = \boldsymbol{p}a_1\boldsymbol{r}_1 + \boldsymbol{p}a_2\boldsymbol{s}_1 + \boldsymbol{p}a_3\boldsymbol{t}_1$$

From (2):

$$\hat{x} = a_1x_1 + a_2x_2 + a_3x_3$$

Similarly:

$$\hat{y} = b_1y_1 + b_2y_2 + b_3y_3$$

Therefore, an image of an object undergoing rotation in 3D space is a linear combination of three basis images.

To consider scaling and translation, let the fourth image, $\hat{\boldsymbol{P}}$ be further transformed by a scale factor, s and the translation vector (t_x, t_y). For any $\boldsymbol{p} \in \boldsymbol{O}$:

$$\hat{\boldsymbol{p}} = (\hat{x}, \hat{y}) = (s\boldsymbol{p}\boldsymbol{u}_1 + t_x, s\boldsymbol{p}\boldsymbol{u}_2 + t_y) \tag{4}$$

Points $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, and $\boldsymbol{p}_3$ remain as defined in (2).

Claim: If both sets $\{\boldsymbol{r}_1, \boldsymbol{s}_1, \boldsymbol{t}_1\}$ and $\{\boldsymbol{r}_2, \boldsymbol{s}_2, \boldsymbol{t}_2\}$ are linearly independent, then there exist scalars a_1, a_2, a_3, a_3, b_1, b_2, b_3, and b_4 such that for every $\boldsymbol{p} \in \boldsymbol{O}$ it holds that:

$$\begin{aligned} \hat{x} &= a_1x_1 + a_2x_2 + a_3x_3 + a_4 \\ \hat{y} &= b_1y_1 + b_2y_2 + b_3y_3 + b_4 \end{aligned}$$

Proof: From above, it holds that there exist scalars c_1, c_2, and c_3 such that:

$$\boldsymbol{u}_1 = c_1\boldsymbol{r}_1 + c_2\boldsymbol{s}_1 + c_3\boldsymbol{t}_1. \tag{5}$$

Since

$$\hat{x} = s\boldsymbol{p}\boldsymbol{u}_1 + t_x,$$

substituting (5) gives

$$\hat{x} = s\boldsymbol{p}c_1\boldsymbol{r}_1 + s\boldsymbol{p}c_2\boldsymbol{s}_1 + s\boldsymbol{p}c_3\boldsymbol{t}_1 + t_x.$$

From (2), (4), and letting

$$\begin{aligned} a_1 &= sc_1 \\ a_2 &= sc_2 \\ a_3 &= sc_3 \\ a_4 &= t_x \end{aligned}$$

it follows that

$$\hat{x} = a_1x_1 + a_2x_2 + a_3x_3 + a_4.$$

Similarly, for $\hat{y}$,

$$\hat{y} = b_1 y_1 + b_2 y_2 + b_3 y_3 + b_4.$$

Therefore, an image of an object undergoing rigid transformations in 3D space is a linear combination using four coefficient pairs.

Ullman and Basri's proofs were based on 3D triplet representations of points. They also state that 4D homogeneous coordinate representations and transformations require four model images, three of which must be different, and a fourth which can be derived from the other three (13).

Note that in the proofs, the determination of x and y coordinates is done independently. This distinction is important, as it results in different coefficients, a_i and b_i, in each dimension. As a consequence, there exists a large system of simultaneous equations that must be satisfied to determine orientation from the coefficients.

A direct result of the creation of a new image from basis images is that only a small number of images need to be stored, rather than an entire 3D model. An additional result is that the linear combination technique can be used for recognition. Suppose that instead of generating a new image, a new image consists of the observation of an object. If the new image can be expressed as the linear combination of the basis images, and the coefficients a_i and b_i meet certain criteria, then the new image is an instance of the object represented in the basis images.

The necessary conditions for recognition by the coefficients are:

$$\begin{gathered}
\| \, a_1 \boldsymbol{r}_1 + a_2 \boldsymbol{s}_1 + a_3 \boldsymbol{t}_1 \, \| = 1 \\
\| \, b_1 \boldsymbol{r}_2 + b_2 \boldsymbol{s}_2 + b_3 \boldsymbol{t}_2 \, \| = 1 \\
(a_1 \boldsymbol{r}_1 + a_2 \boldsymbol{s}_1 + a_3 \boldsymbol{t}_1)\,(b_1 \boldsymbol{r}_2 + b_2 \boldsymbol{s}_2 + b_3 \boldsymbol{t}_2) = 0
\end{gathered} \tag{6}$$

Ullman and Basri propose three methods for determining the coefficients each designed to minimize the difference between the unknown view and the view generated from the linear combination. One method employs corresponding features in both the model and the image.

The second involves an iterative search of the coefficient space. Lastly, the third method uses linear receptive fields to map different views of the same object into a common representation.

In this thesis, the first method was used. Conspicuous vertices of the model are used as alignment points. The transformation of each of these points from the basis images to their corresponding location in the unknown view is a function of the coefficients.

The $x-$ and $y-$coordinates of the alignment points are arranged in two vectors, $\boldsymbol{p}_x$ and $\boldsymbol{p}_y$. The $x-$ and $y-$coordinates of the corresponding points in the basis images are arranged into two basis matrices, $\boldsymbol{X}$ and $\boldsymbol{Y}$. Vectors of the coefficients, $\boldsymbol{a}$ and $\boldsymbol{b}$, were then found analytically by direct algebraic manipulation of the image matrices. This was possible because of the small number of vertices used as alignment points for attitude determination. Specifics of the approach taken are discussed in Chapter III.

2.5 Previous Applications

2.5.1 Recognition. Ullman and Basri have used the linear combination approach in recognition of objects. Their methods include algorithms that minimize the difference between the image of the object to be recognized and a view generated from the linear combination of basis images. In one method, called full alignment, distortions are allowed so that the LC image matches exactly. The solution is then evaluated on the plausibility of the transformation. If the model is distorted too much, then the object of concern is not represented by the model. In the minimal alignment technique, only rigid transformations are allowed. A close match after rigid transformations indicates recognition. Ullman and Basri used a third method that does not rely on matching images and features. The method, called alignment search, searches the space of possible transformations. If a satisfactory transformation in the allowable set is found, then the object is recognized.

Thau combined linear combinations and neural networks for recognition (12). In his experiments, inputs based on a 2D image of the object to be recognized were fed into the neural network. The network was trained beforehand with appropriate basis views of an object. The neural net performed all the calculations, and output the confidence of recognition. The coefficients are never explicitly found. Thau also suggests that the neural network can be configured to output attitude and position information as well. The utility of this concept and the circumstances in which this might be a good idea will be discussed further in Chapter III.

2.5.2 Image Registration. In image registration, the basic goal is the alignment of points between separate data sets, then transforming each data set so that each alignment point is in the same position in each image. Image registration can be performed in 3D and 2D.

In the 3D case, Rizuto, as a component of his 1991 thesis, *Three-Dimensional Medical Image Registration Using a Patient Space Correlation Technique*, sought to determine geometric transformations required to orient two volumes so that user selected anatomic landmarks were aligned (5).

In a collateral thesis, *Identity Verification Through the Use ofFace Recognition and Speaker Identification*, Keller registers facial features, such as eyes, to specific locations in a 2D image frame (6).

In both efforts mentioned, the composite transformation was found, then used to transform a volume or image as necessary. In neither case, however, was it necessary to decompose the transformation matrix into the actual transformation parameters such as rotation, translation, or scaling. The extraction of the transformation parameters from the transformation matrix is the goal of this thesis.

III. Linear Combination for Attitude Determination

3.1 Introduction

This chapter describes the specific techniques employed in this thesis. It begins by discussing the generalization of the linear combination theory to encompass 4D homogeneous coordinates and all rigid transformations. Methods for determining coefficients and extracting orientation information are then presented. Modeling and testing of the technique in *Mathematica* and implementation on the Silicon Graphics workstation are also presented.

3.2 Linear Combination using 4D Homogeneous Coordinates

3.2.1 Matrix Representation of Images. Throughout this effort, images are represented and manipulated in the form of matrices. An image object can be represented by a matrix, $\boldsymbol{P}$, which contains the $x-$ and $y-$coordinates (if the image is an xy projection) of important vertices of the object in that particular image. Several images of a single object may exist. The ith image of an object with n vertices, $\boldsymbol{P}_i$ then takes the form of

$$\boldsymbol{P}_i = \begin{bmatrix} x_{i,1} & y_{i,1} \\ x_{i,2} & y_{i,2} \\ x_{i,3} & y_{i,3} \\ & \vdots & \\ x_{i,n} & y_{i,n} \end{bmatrix}$$

Note that the vertices are ordered. While the order is arbitrary, it must be consistent throughout all images of the same object. That is, if the vertex representing the lower left corner of the right solar panel of a particular satellite appears as the fifth element (row) in one matrix representation of an image, then the fifth row in all matrix images of the same satellite must represent the same (lower left) corner of the same (right) solar panel.

3.2.2 Basis Matrices. As discussed in Chapter II, the linear combination of several basis images to form a new image can be expressed in matrix form. The column vectors of the new image, $\boldsymbol{p}_x$ and $\boldsymbol{p}_y$, are calculated by

$$\boldsymbol{p}_x = \boldsymbol{X}\boldsymbol{a}$$

$$\boldsymbol{p}_y = \boldsymbol{X}\boldsymbol{b}.$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are column vectors of the coefficients of the linear combination, and $\boldsymbol{X}$ and $\boldsymbol{Y}$ are matrices of the first and second columns of the basis images, respectively.

The $\boldsymbol{X}$ and $\boldsymbol{Y}$ matrices formed from m basis images then have the following forms.

$$\boldsymbol{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \cdots & x_{1,m} \\ x_{2,1} & x_{2,2} & x_{2,3} & \cdots & x_{2,m} \\ x_{3,1} & x_{3,2} & x_{3,3} & \cdots & x_{3,m} \\ & & \vdots & & \\ x_{n,1} & x_{n,2} & x_{n,3} & \cdots & x_{n,m} \end{bmatrix}$$

and

$$\boldsymbol{Y} = \begin{bmatrix} y_{1,1} & y_{1,2} & y_{1,3} & \cdots & y_{1,m} \\ y_{2,1} & y_{2,2} & y_{2,1} & \cdots & y_{2,m} \\ y_{3,1} & y_{3,2} & y_{3,3} & \cdots & y_{3,m} \\ & & \vdots & & \\ y_{n,1} & y_{n,2} & y_{n,3} & \cdots & y_{n,m} \end{bmatrix}$$

where $x_{n,m}$ and $y_{n,m}$ are the $x-$ and $y-$coordinates of the nth vertex in the mth basis image, respectively. The dimensions of $\boldsymbol{X}$ and $\boldsymbol{Y}$ are thus dependent on the number of basis images and the number of vertices in each image, and the coefficient vectors, $\boldsymbol{a}$ and $\boldsymbol{b}$ are equal in length to the number of basis images.

The coefficients are then found by

$$\boldsymbol{a} = \boldsymbol{X}^{+}\boldsymbol{p}_x \tag{7}$$

and

$$\boldsymbol{b} = \boldsymbol{Y}^{+}\boldsymbol{p}_y \tag{8}$$

where $\boldsymbol{X}^{+}$ and $\boldsymbol{Y}^{+}$ are the pseudo inverses (or inverses if the matrices are square) of $\boldsymbol{X}$ and $\boldsymbol{Y}$. The existence of these inverses can be guaranteed by careful selection of the basis images. Choosing basis images is discussed in the next subsection.

A complicated object may have a great many vertices, leading to very large matrix representations. For determination of alignment, however, only a small subset of these vertices are needed. In general, the choice of these vertices is arbitrary. These vertices become the alignment points. The alignment points should represent conspicuous features that are visible at most orientations. For simplicity in finding inverses, m and n are chosen so that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are square, that is, the number of alignment points equals the number of basis images.

3.2.3 Choosing Basis Images. In order to avoid singularity of the $\boldsymbol{X}$ and $\boldsymbol{Y}$ matrices (ensuring the existence of their inverses), care must be taken so that the columns of each matrix are independent. If a set of basis images results in $\boldsymbol{X}$ and $\boldsymbol{Y}$ with independent columns, then the basis images can be thought of as being independent themselves.

As discussed in Chapter II, three basis images are necessary to represent an object that has undergone any combination of rotation and scaling. If rotation and scaling are the only transformations being considered, then a system of three images with three alignment points each will be sufficient. Basis images are easily chosen. Almost any three images of the object at three different rotations will be independent. In some cases, however, singular matrices arise, such as when the first image has no rotation; the second, a pure rotation about the $x-$axis; and the third, a pure rotation about the $y-$axis (13).

If translations are to be considered, the choice of basis images becomes more complicated. In theory, four images are needed. However, because exactly three images span the scaled and rotated image space of an object, that is, describe all scaling and rotation, no more than three images that represent only scaling and rotation can be independent. In other words, the basis image set must include translations in both x and y if four basis images are to be used. However, the x and y translations need not occur in the same image.

In practice, three basis images without translation are sufficient. Used in the place of the fourth image is a contrivance that greatly simplifies the extraction of the translation transformation parameters. This concept is discussed in the next section.

The system developed in this thesis uses three basis images with four alignment points each, and considers all rigid transformations of an object. When presented with an image of an object at an unknown orientation, the coefficients for combining the basis images to form the unknown image can be found.

The coefficients can then be tested against the constraints outlined in Chapter II (Equation 6). If the coefficients satisfy the constraints, then it can be said that the object in the unknown image is an instance of the object represented by the basis images.

3.3 Determining Alignment Parameters From the Coefficients

Once it is determined that the object in the unknown image is in fact an instance of the object in the basis images, one can proceed to extract a set of transformations that will result in the pose shown in the unknown image. In practice, there may be an infinite number of transformations that result in the same pose. Rotations, for instance, can occur either clockwise or counter clockwise. The order of transformations is important as well. A rotation of 23° about the $x-$axis followed by a rotation of 49° about the $z-$axis yields one result, while a rotation of 49° about the $z-$axis followed by a rotation of 23° about the $x-$axis yields another.

The transformations must be determined relative to a specific orientation through a defined sequence of operations. For the purposes of this thesis, the baseline orientation is that of a stored 3D model. The specific sequence of the transformations is scaling, rotation about the $x-$axis, rotation about $y-$axis, rotation about $z-$axis, and lastly, translation. All rotations are measured counter-clockwise.

With the proper choice of basis images, translation can be extracted directly from the coefficients. The $x-$coordinates of the alignment points in the new image are in the first column of $\hat{\boldsymbol{P}}$, which is referred to by $\hat{\boldsymbol{p}}_{\boldsymbol{x}}$. Expanding (7) gives

$$\hat{\boldsymbol{p}}_{\boldsymbol{x}} = a_1\boldsymbol{p}_{\boldsymbol{x},1} + a_2\boldsymbol{p}_{\boldsymbol{x},2} + a_3\boldsymbol{p}_{\boldsymbol{x},3} + a_4\boldsymbol{p}_{\boldsymbol{x},4}.$$

Since three terms are sufficient to represent all scaling and rotation, the first three terms can be arbitrarily assigned to represent scaling and rotation, and the fourth term assigned to represent translation. In this case, translation is merely the shifting of the scaled and rotated object by t_x.

$$\hat{\boldsymbol{p}}_x = c_1\boldsymbol{p}_{x,1} + c_2\boldsymbol{p}_{x,2} + c_3\boldsymbol{p}_{x,3} + t_x$$

By choosing $\boldsymbol{p}_{x,4} = \boldsymbol{1}$, if follows that $t_x = a_4$. Similar manipulation of (8) will yield $t_y = b_4$. This means that if the fourth basis image, $\boldsymbol{P}_4$, is replaced by a vector of 1's, and the remaining basis images $\boldsymbol{P}_i$, for $i \in \{1,2,3\}$, are formed from pure scaling and rotation, the fourth coefficients, a_4 and b_4, will be the amounts of translation in x and y, respectively. The $\boldsymbol{X}$ and $\boldsymbol{Y}$ matrices thus take the form of

$$\boldsymbol{X} = \begin{bmatrix} x_{1,1} & x_{2,1} & x_{3,1} & x_{4,1} \\ x_{1,2} & x_{2,2} & x_{3,2} & x_{4,2} \\ x_{1,3} & x_{2,3} & x_{3,3} & x_{4,3} \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

and

$$\boldsymbol{Y} = \begin{bmatrix} y_{1,1} & y_{2,1} & y_{3,1} & y_{4,1} \\ y_{1,2} & y_{2,2} & y_{3,2} & y_{4,2} \\ y_{1,3} & y_{2,3} & y_{3,3} & y_{4,3} \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Note that with careful choice of the basis images, the first three row vectors are independent. The fourth row is independent to the other three, thus both matrices are 4 × 4 square and non-singular. That is, their inverses exist.

Extraction of the scaling and rotation transformations is a bit more complicated, and requires a 3D reference model to be calculated exactly. It involves recovering the first two columns of the original composite transformation matrix $\boldsymbol{T}_c$, as defined in (1). That is, the composite matrix that transforms the 3D reference model to the pose represented in the unknown image. Values for the individual transformations, in the sequence specified above, that result in the composite matrix can then be found.

Substituting matrix inverses $\boldsymbol{X}^{-1}$ and $\boldsymbol{Y}^{-1}$ for pseudoinverses $\boldsymbol{X}^{+}$ and $\boldsymbol{Y}^{+}$ in (7) and (8), the coefficients for the combination of x–coordinates are

$$\boldsymbol{a} = \boldsymbol{X}^{-1}\boldsymbol{p}_x \tag{9}$$

and the coefficients for the combination of y–coordinates are

$$\boldsymbol{b} = \boldsymbol{Y}^{-1}\boldsymbol{p}. \tag{10}$$

Substituting for $\boldsymbol{p}_x$,

$$\boldsymbol{a} = \boldsymbol{X}^{-1}\boldsymbol{O}\boldsymbol{u}_x$$

where $\boldsymbol{O}$ is the 3D reference model of the form

$$\boldsymbol{O} = \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{bmatrix}$$

and $\boldsymbol{u}_x$ is the first column of the composite transformation matrix. Solving for $\boldsymbol{u}_x$,

$$\boldsymbol{u}_x = \boldsymbol{O}^{-1}\boldsymbol{X}\boldsymbol{a}. \tag{11}$$

Similarly, $\boldsymbol{u}_y$, the second column of the composite transformation matrix can be found by

$$\boldsymbol{u}_y = \boldsymbol{O}^{-1}\boldsymbol{Y}\boldsymbol{b}. \tag{12}$$

Alternatively, if the 3D reference model $\boldsymbol{O}$ is not available, $\boldsymbol{u}_x$ and $\boldsymbol{u}_y$ can be calculated from the linear combination coefficients and the transformations used to generate the basis images. Since

$$\hat{x} = \boldsymbol{p}\boldsymbol{u}_1 = \boldsymbol{p}a_1\boldsymbol{r}_1 + \boldsymbol{p}a_2\boldsymbol{s}_1 + \boldsymbol{p}a_3\boldsymbol{u}_1 + \boldsymbol{p}a_4\boldsymbol{w}_1,$$

it follows that

$$\boldsymbol{u}_1 = a_1\boldsymbol{r}_1 + a_2\boldsymbol{s}_1 + a_3\boldsymbol{u}_1 + a_4\boldsymbol{w}_1$$

and

$$\boldsymbol{u}_2 = b_1\boldsymbol{r}_2 + b_2\boldsymbol{s}_2 + b_3\boldsymbol{u}_2 + b_4\boldsymbol{w}_2.$$

Thus, the transformation matrix $\boldsymbol{u}$ is a linear combination of the transformation matrices $\boldsymbol{r}$, $\boldsymbol{s}$, $\boldsymbol{t}$ and $\boldsymbol{w}$ used to generate the basis images. For the choice of basis images outlined above,

$$\boldsymbol{w} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Basis image transformation matrices, $\boldsymbol{r}$, $\boldsymbol{s}$, and $\boldsymbol{t}$, can be calculated since the pose of the object in the basis images is known.

Using this alternative technique means that, as a minimum, only three 2D basis images and their pose information needs to be stored beforehand. Having the entire 3D model is helpful, however, since the coefficients can be calculated using simple matrix multiplication.

Each element of the $\boldsymbol{u}_x$ and $\boldsymbol{u}_y$ are functions of the transformation parameters: scaling, s; rotation, θ_x, θ_y, θ_z; and translation t_x, t_y. They are related by the following equations:

$$\begin{aligned} u_{x,1} &= s\cos\theta_y\cos\theta_z \\ u_{x,2} &= s(\cos\theta_z\sin\theta_x\sin\theta_y - \cos\theta_x\sin\theta_z) \\ u_{x,3} &= s(\cos\theta_x\cos\theta_z\sin\theta_y + \sin\theta_x\sin\theta_z) \\ u_{x,4} &= t_x \end{aligned} \tag{13}$$

$$\begin{aligned} u_{y,1} &= s\cos\theta_y\sin\theta_z \\ u_{y,2} &= s(\sin\theta_x\sin\theta_y\sin\theta_z + \cos\theta_x\cos\theta_z) \\ u_{y,3} &= s(\cos\theta_x\sin\theta_y\sin\theta_z - \cos\theta_z\sin\theta_x) \\ u_{y,4} &= t_y \end{aligned} \tag{14}$$

Again, the last elements, $u_{x,4}$ and $u_{y,4}$, are easily recognized as the translation, t_x and t_y because of the choice of basis images. Because of the restriction of equal scaling along each axis, the other elements and equations form a system of six non-linear equations and four unknowns. The parameters of the composite transformation can be found by solving this system of equations using an iterative numerical method such as Newton's method or gradient descent.

Recognizing that $\boldsymbol{u}_x$ and $\boldsymbol{u}_y$ are columns of a transformation matrix leads to further simplification of (13) and (14). The submatrix composed of the upper left 3×3 matrix of $\boldsymbol{T}_c$ is the transformation matrix for scaling and rotation. Such matrices have the property that the rows and columns are orthogonal. Further, if the matrix represents only rotations, then the rows and columns are orthonormal. Scale factors in the x and y directions can then be found by

$$s_x = \sqrt{u_{x,1}^2 + u_{x,2}^2 + u_{x,3}^2} \tag{15}$$

and

$$s_y = \sqrt{u_{y,1}^2 + u_{y,2}^2 + u_{y,3}^2}. \tag{16}$$

Since rigid scaling has been assumed, then

$$s = s_x = s_y. \tag{17}$$

The above calculations reduce (13) and (14) to

$$\begin{aligned} u'_{x,1} &= \cos\theta_y \cos\theta_z \\ u'_{x,2} &= \cos\theta_z \sin\theta_x \sin\theta_y - \cos\theta_x \sin\theta_z \\ u'_{x,3} &= \cos\theta_x \cos\theta_z \sin\theta_y + \sin\theta_x \sin\theta_z \end{aligned} \tag{18}$$

and

$$\begin{aligned} u'_{y,1} &= \cos\theta_y \sin\theta_z \\ u'_{y,2} &= \sin\theta_x \sin\theta_y \sin\theta_z + \cos\theta_x \cos\theta_z \\ u'_{y,3} &= \cos\theta_x \sin\theta_y \sin\theta_z - \cos\theta_z \sin\theta_x \end{aligned} \tag{19}$$

where $u'_{a,n} = \frac{u_{a,n}}{s}$, $a = x, y$, and s is as defined in (17).

The above equations, (18) and (19), describe two systems of three equations and three unknowns with identical solutions. Each system, however, has multiple solutions, and are not easily solved by numerical methods. Analytical solutions may exist, as described later. Consequently, (13) and (14) are used to determine the alignment parameters.

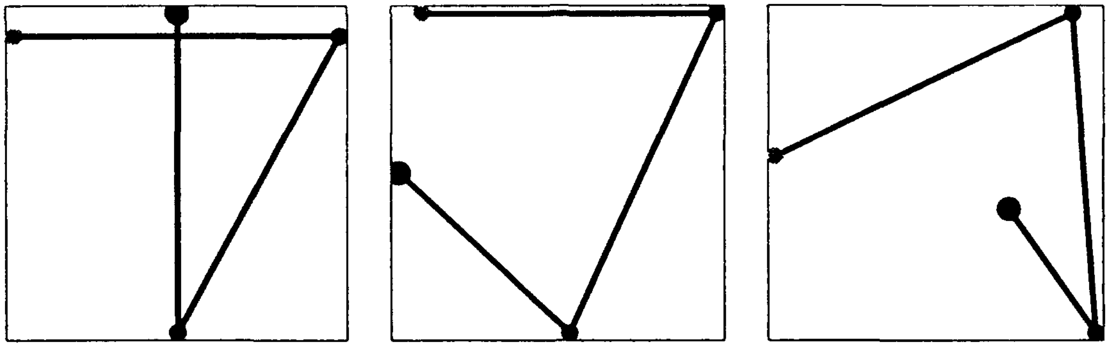


Figure 5. Basis images of *four*.

3.4 Modeling in Mathematica

The technique described above was implemented in *Mathematica*. *Mathematica's* notebook environment provided an ideal vehicle for setting up and exercising the method on a simple object.

The object chosen for testing is shown in Figure 1. It is simply a collection of four ordered vertices connected by line segments. Images of the object are orthographic projections onto the xy plane. The object is called *four*, both because it has four vertices, and because certain projections take the shape of the numeral four.

The basis images are projections of the object rotated by 30° about an axis. In total, three basis images are used, each with a rotation about a single axis. The basis images are shown in Figure 5.

The computer determines a pose for the object by generating random values for scaling, rotation about three axes, and translation in x and y. Translations in z will not be apparent in an orthographic projection. The transformations are combined to form a composite transformation matrix. Finally, the object is transformed and projected to create an image of the object at the random pose. This image is then used as a test of the pose determination system. An example of *four* at a randomly generated pose is shown in Figure 6.

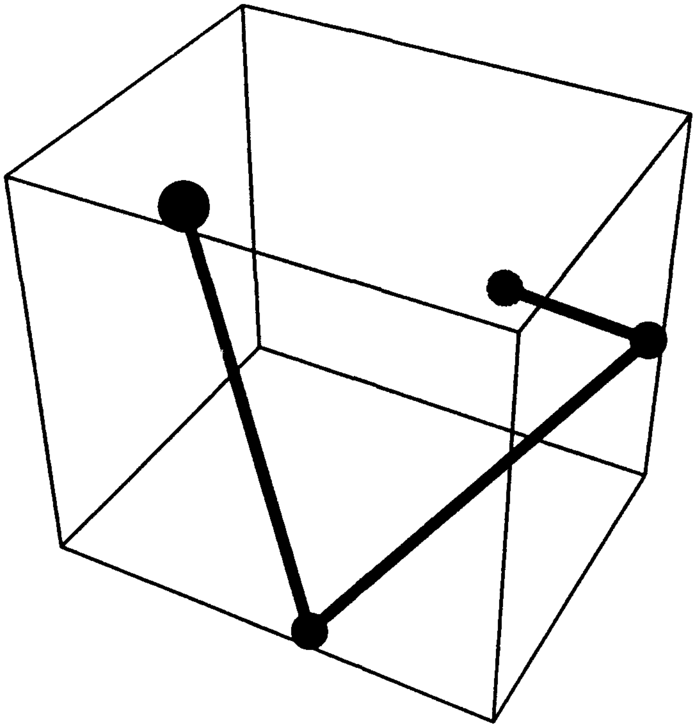


Figure 6. Random transformation of the object *four*.

The basis matrices, $\boldsymbol{X}$ and $\boldsymbol{Y}$, are then extracted from the basis images (Figure 5)for calculation of the coefficients. Calculations of $\boldsymbol{a}$ and $\boldsymbol{b}$ (the $x-$ and $y-$coefficients) are done separately. Once calculated, the fourth elements of both $\boldsymbol{a}$ and $\boldsymbol{b}$ can be checked against the translation values of the transformation parameters. These values should be the same. Additionally, the coefficients can be used to generate a new image from the basis images. The new image should match the unknown image exactly.

From the coefficient vectors, the first two columns of the composite transformation matrix, $\boldsymbol{u}_x$ and $\boldsymbol{u}_y$, are then calculated by (11) and (12). Newton's method is then used to solve the system of equations (13) and (14). Note that only six equations are used: $u_{x,4}$ and $u_{y,4}$ are disregarded. The roots of the system are the parameters for scaling and rotation about each axis, s, θ_x, θ_y, and θ_z. Recall that the translation parameters, t_x and t_y, are found analytically from a and b, and Newton's method is not needed to find them. The scaling parameter, s can be checked using equations (15), (16), and (17).

In *Mathematica*, Newton's method is limited to having the same number of equations and unknowns. The system under consideration has six equations and four unknowns. Working within this limitation, only four equations are input to Newton's method. As a result, *Mathematica* finds a solution to the four equations input, which is not always a solution to all six equations. Convergence to erroneous roots is minimized by the choice of initial guesses input to *Mathematica*. The initial guesses are random numbers between 1 and 2 for the scaling, s, and between 0 and π for the angles of rotation, θ_x, θ_y, and θ_z. When *Mathematica* finds an erroneous solution, Newton's method can simply be rerun with newly generated random initial guesses.

Correctly solving the system of equations results in a complete set of the transformations necessary to recreate the pose in the unknown image. As mentioned before, the set of transformations is not unique. Rotations may be clockwise instead of counter clockwise, or have extraneous factors of 2π. When performed in the correct order (scaling; rotation in x, y, then z; and translation) the original object, *four*, can be transformed and projected to duplicate the unknown image. Figure 7 shows an original, randomly generated, unknown image, and the duplicate image generated from the transformation determined by the linear combination technique. The images match exactly.

3.5 Implementation on Silicon Graphics Workstation

A modular approach was used in the implementation of the alignment system on the SGI. Figure 8 shows the basic processing architecture of the system. User interface functions are not explicitly shown here, but are presented in detail in Appendix A: Users Guide.

Peripheral components of the system include the 3D model and the Model Data Manipulator. These components provide data in the required format to the main processing component. The main processing component of the system is the coefficient and parameter extraction block. It consists of a separate program called `matcher`. It is in turn composed of

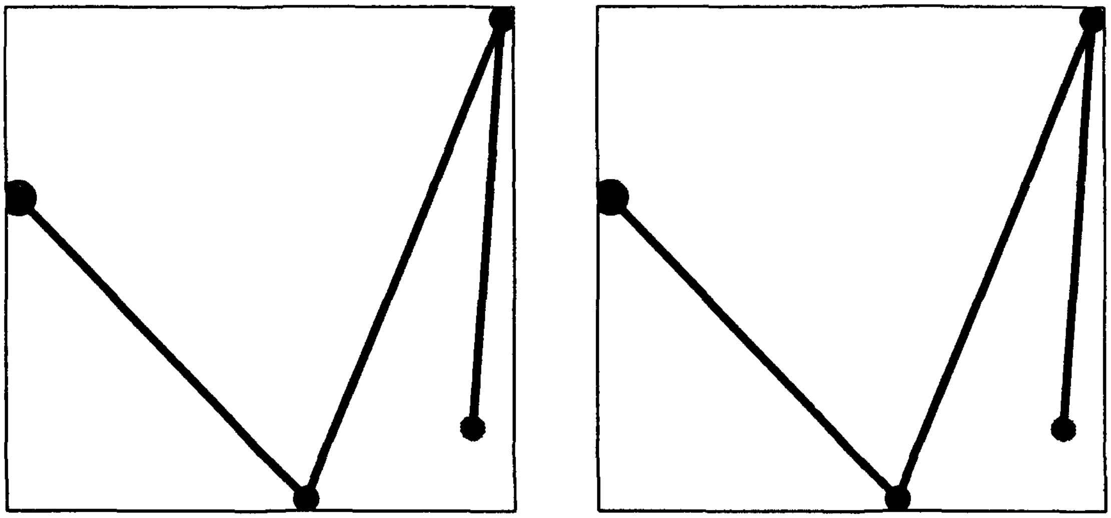


Figure 7. Random and matched images.

modules for calculating the coefficients, transform matrix, and transformation parameters. Each component is discussed in the following subsections.

3.5.1 3D Model. The 3D model is stored in a modified New Solid Model (NSM) format. The NSM format is generated by the SMT satellite modeling program used by the Philips Laboratory Satellite Assessment Center (2). The data is stored in a hierarchical structure. Simple assemblies, or model elements, are composed of eleven primitives including rectangular parallelepipeds, boxes, wedges, hexahedrons, cones, elliptical cylinders, elliptical cones, spheres, hemispheres and ellipsoids. More complex assemblies are then the model elements or other assemblies. Finally the entire satellite is represented by a few rather complex assemblies. NSM format files are customarily given the `.nsm` suffix. A slight modification is made to the NSM format so that alignment point information is included in the model file. The modified format is distinguished by the suffix `.nsm.alp`. Details of the file format are covered in the User's Guide (Appendix A).

3.5.2 Model Data Manipulation. The Model Data Manipulator generates images for use in the parameter extraction program. It reads the `.nsm.alp` model files and writes

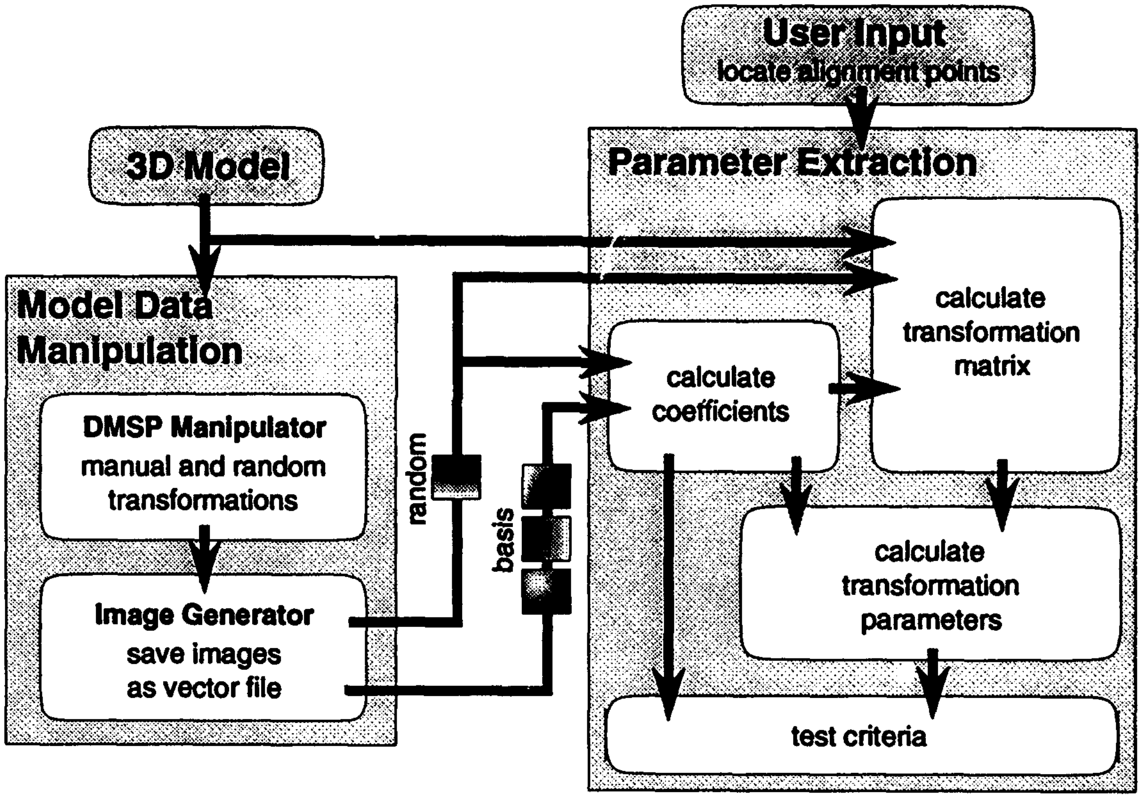


Figure 8. Parameter extraction processing architecture

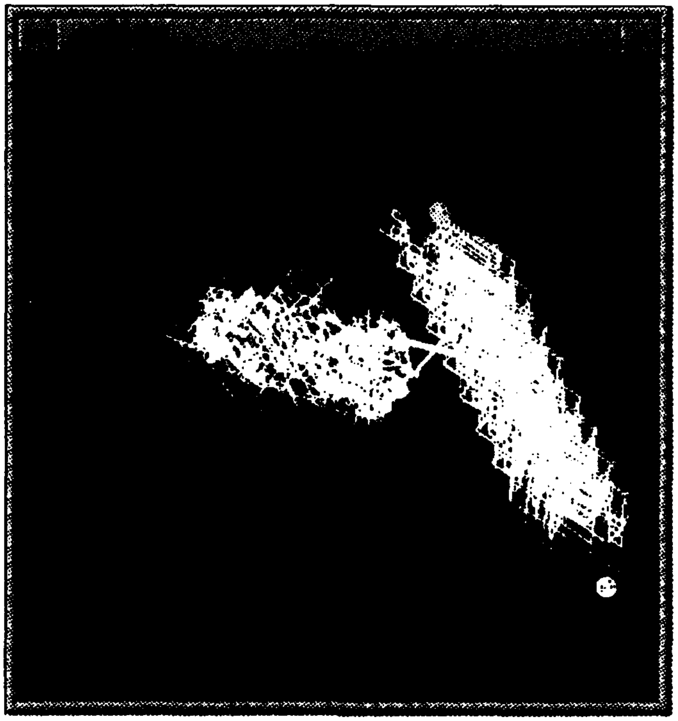


Figure 9. Image generator window on the SGI workstation.

the vector image files used as basis images and simulated sensor images. It is embodied in the program `writesat`. Executing `writesat` is covered in the User's Guide.

The procedure used by `writesat` is easy to follow. It reads the model file, and builds a hierarchical structure in memory representing the satellite. It then traverses the structure to render a GL (SGI Graphics Library) graphical object in wireframe. Graphical objects are a collection of drawing commands that can be rapidly executed and manipulated by the SGI workstation (8). The prescribed transformations are then loaded into the SGI's internal transformation matrix.

The graphical object is transformed, projected and displayed on the screen. Alignment points appear as colored spheres superimposed on the image. An example of the `writesat` window is shown in Figure 9. The user then has the opportunity to modify the pose before creating an output file. As the pose is modified, the current transformation parameters are output to the screen.

When a satisfactory view is achieved, `writesat` creates the output file. First, the transformation matrix is retrieved from the SGI's matrix stack. Then the hierarchical structure is traversed, transforming each vertex of every line as the entire structure is rendered. The vertex pairs that define each line are written to the output file, resulting in a vector file format rather than a pixel based image file. The header of the file contains the locations of the alignment points and the transformation parameters used to create the view.

3.5.3 Parameter Extraction. The `matcher` program is the main component of the attitude determination system. Its inputs include the 3D satellite model, three basis images, the simulated sensor image, and user designation of the alignment points. Outputs include coefficients for linear combination of the basis images and alignment parameters. Additionally, it calls routines that display all the images and an orthographic projection of the 3D model. Once alignment parameters are found, the 3D model is transformed and projected. The quality of the alignment parameters can then be checked by comparing the projection of the transformed model and the sensor image.

The `matcher` program is launched from a C shell window and requires a control file that specifies each basis image, the sensor image, and the 3D model. Execution of `matcher` is covered in the User's Guide.

Each image input file is in the vector format produced by the `writesat` program. As discussed above, the vector files are 3D objects. The files specified as images are automatically projected to an orthographic view as they are read in, converting them to true 2D images. The 3D model file is read in without modification. It is used to build a SGI graphical object, which is in turn manipulated and projected using GL routines. The image display and manipulation functions are written as separate subroutines that work specifically with the vector image format. Modifications of the code for use with pixel formatted images should be a simple matter of replacing these routines with appropriate image handling routines.

Upon launching, the `matcher` program opens windows that display each basis image, the simulated sensor image and the 3D model (if available). An example of the `matcher` screen is illustrated in Figure 10.

Note that because the basis and sensor images are 2D, the alignment points are represented as crosshairs, rather than spheres as in the `writesat` program. The 3D model display retains the spherical alignment points so that they will be visible at any pose.

After the user has identified the alignment points in the sensor image, the coefficients for linear combination or the transformation parameters may be calculated. Separate modules are used for calculation of the coefficients and the transformation parameters so that calculation of the transformation parameters is not necessarily dependent on calculation of the coefficients.

The calculations rely heavily on routines from *Numerical Recipes in C* (10). Coefficients for the linear combination are calculated as described in the previous section. Equations (7) and (8) are solved using lower-upper decomposition (`ludcmp`) and back substitution (`lubksub`) routines from the *Numerical Recipes* library.

Lower-Upper (LU) Decomposition with backsubstitution is a method of solving linear systems of equations. To solve for the coefficients of the x−coordinates, $\boldsymbol{a}$, (9) is rewritten as

$$\boldsymbol{X}^{-1}\boldsymbol{p}_x = \boldsymbol{L}\boldsymbol{U}\boldsymbol{p}_x = \boldsymbol{a},$$

where

$$\boldsymbol{L} = \begin{bmatrix} \alpha_{1,1} & 0 & 0 & 0 \\ \alpha_{2,1} & \alpha_{2,2} & 0 & 0 \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & 0 \\ \alpha_{4,1} & \alpha_{4,2} & \alpha_{4,3} & \alpha_{4,4} \end{bmatrix}$$

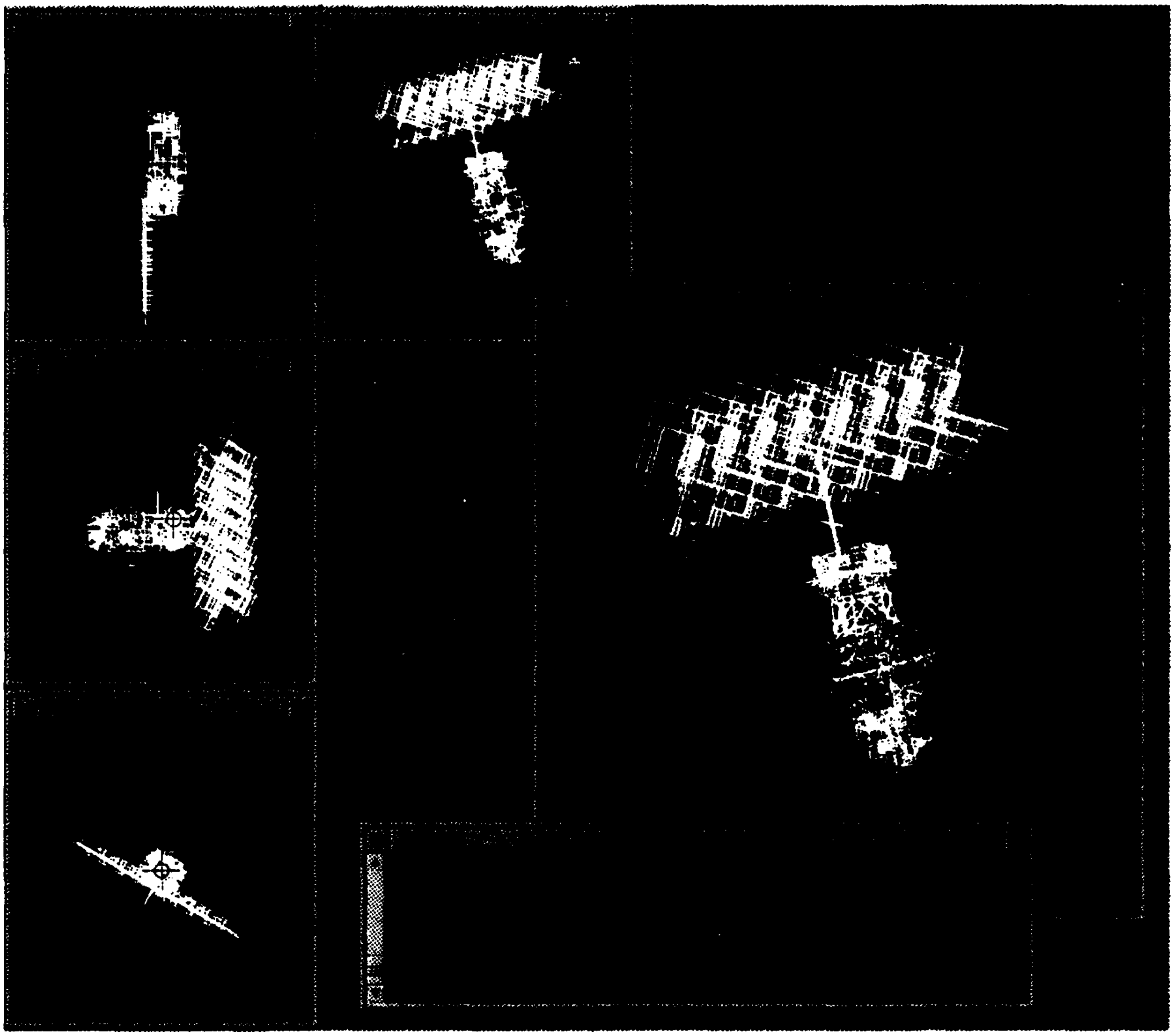


Figure 10. Parameter extraction windows on the SGI workstation.

and

$$\boldsymbol{U} = \begin{bmatrix} \beta_{1,1} & \beta_{1,2} & \beta_{1,3} & \beta_{1,4} \\ 0 & \beta_{2,2} & \beta_{2,3} & \beta_{2,4} \\ 0 & 0 & \beta_{3,3} & \beta_{3,4} \\ 0 & 0 & 0 & \beta_{4,4} \end{bmatrix}$$

and then solving

$$\boldsymbol{L}\boldsymbol{y} = \boldsymbol{a} \tag{20}$$

and

$$\boldsymbol{U}\boldsymbol{p}_x = \boldsymbol{y}. \tag{21}$$

Equation (20) is solved by forward substitution

$$y_1 = \frac{p_{x,1}}{\alpha_{11}}$$

$$y_i = \frac{1}{\alpha_{ii}}\left[p_{x,i} - \sum_{j=1}^{i-1} \alpha_{ij} y_j\right] \quad i = 2, 3, \ldots, N$$

while (21) can then be solved by backsubstitution

$$a_N = \frac{y_N}{\beta_{NN}}$$

$$a_i = \frac{1}{\beta_{ii}}\left[y_i - \sum_{j=i+1}^{N} \beta_{ij} a_j\right] \quad i = N-1, N-2, \ldots, 1.$$

The process is repeated for the y coefficients, $\boldsymbol{b}$, using (10).

The calculation of the transformation matrix, $\boldsymbol{T}_c$, varies slightly from that described above. Rather than using (11) and (12), $\boldsymbol{T}_c$ is found by

$$\boldsymbol{T}_c = \boldsymbol{O}^{-1}\hat{\boldsymbol{P}} \tag{22}$$

where $\boldsymbol{O}$ and $\hat{\boldsymbol{P}}$ are the matrix representations of the 3D model and the unknown image.

The first two columns of $\boldsymbol{T}_c$ are then $\boldsymbol{u}_x$ and $\boldsymbol{u}_y$. The translation parameters, $t_x = u_{x,4}$ and $t_y = u_{y,4}$ can be extracted directly. The elements of $\boldsymbol{u}_x$ and $\boldsymbol{u}_y$ can then be input to an appropriate numerical method with the equations (13) and (14) to find the remaining transformation parameters s, θ_x, θ_y, and θ_z.

The Numerical Recipes `ludcmp` and `lubksub` routines are used to find $\boldsymbol{u}_x$ and $\boldsymbol{u}_y$, then the Newton-Raphson root finding routine `mnewt` with a randomly generated starting solution is used to solve for the alignment parameters.

The Newton-Raphson method is an iterative process for finding the roots of a system of equations. For the one-dimensional case, several values of x_i are tested as roots of the function, $f(x)$. Subsequent trial solutions, x_{i+1} are found by

$$x_{i+1} = x_i + \delta(x_i)$$

where

$$\delta(x) = -\frac{f(x)}{f'_x(x)}.$$

The method is generalized to four dimensions and applied to (13) and (14).

If a solution vector is found, it is checked against (13) and (14). If it is an extraneous root, `mnewt` is rerun with another randomly generated initial guess until a valid root is found. The number of iterations of `mnewt` is returned with the solution.

The numerical routines are written as separate subroutines so that they may be replaced easily in order to implement and test alternate methods. All the major programs used in this thesis are included in a makefile for ease of compilation and linking.

IV. Performance Analysis

This chapter presents an analysis of the performance of the parameter extraction system. For the tests performed, the previously discussed DMSP model was transformed to a random pose and orthographically projected to form the simulated sensor image. The basis images were taken of the satellite rotated 60° about the $x-$, $y-$, and $z-$axes.

A separate program **sens** was written to call the parameter extraction routines repeatedly while varying alignment point errors and tolerances. The desired variations in tolerance and alignment points are input to **sens**, and an ASCII file containing introduced errors, extracted parameters, and error metrics is created as output. Running **sens** is covered in the User's Guide (Appendix A).

In the analysis, the effects of tolerance and alignment point error on the parameter extraction performance were investigated. The locations of alignment points in the reference model and the simulated sensor image are shown in Table 1. The results of the experiments are described below.

4.1 Tolerance

Two tolerances, `tolx` and `tolf` are input to the Numerical Recipes root finding routine `mnewt`. The first, `tolx`, is compared to the difference between the tentative solution

Table 1. Data for performance analysis

	AP No.	x (inches)	y (inches)	z (inches)
Model	1	65.379997	13.509999	-21.000000
	2	-25.040001	127.193787	114.209007
	3	-12.882499	-156.188995	119.519012
	4	0.000000	1.194350	-127.111397
Sensor	1	24.570559	-15.408206	
	2	-137.169266	92.209892	
	3	127.339104	165.023773	
	4	38.865135	-111.773666	

vector (see discussion of Newton-Raphson method in section 3.5.3) in successive iterations. Iterations stops if δ is less than `tolx`. The second, `tolf`, is compared to the functions, here (13) and (14), for which roots are sought. If the values of the functions at the trial solution vector $\boldsymbol{X}$ are less than `tolf`, iteration stops. For simplicity, `tolx` and `tolf` are taken to be equal, and can be varied from the control menu in the parameter extraction program `matcher`. For the case of (13) and (14), `tolx` and `tolf` are unitless.

To illustrate the effect of tolerance on the accuracy of the extracted parameters, The parameter extraction routine was run ten times each at tolerances varying from 0.00001 to 0.1. Each trial resulted in a set of transformation parameters, $\boldsymbol{S}$. $\boldsymbol{S}$ includes s, $\boldsymbol{\theta_x}$, $\boldsymbol{\theta_y}$, $\boldsymbol{\theta_z}$, $\boldsymbol{t_x}$, and $\boldsymbol{t_y}$ as previously defined. The 3D model of the satellite is then transformed using $\boldsymbol{S}$, and the positions of the alignment points compared with their true locations, that is, the locations that resulted from the pose depicted in the simulated sensor image. The scatter plot in Figure 11 shows the log-rms errors across all four alignment points in x, y, and z for each trial. Clearly, the error increases roughly linearly as the tolerance increases. Note that the units of the plot are log(inches), and the errors are in fact quite small given that the satellite's size is on the order of hundreds of inches. The extracted transformation parameters are shown in Figures 12 through 16.

Immediately obvious from Figure 12 is the bimodal nature of the rotation parameters, despite the fixed order of transformation. There are two distinct sets of rotations, denoted by I and II in the figure, that result in the correct pose. By inspection, the following relationship between I and II can be deduced.

$$\begin{aligned} \alpha_{II} &= \alpha_I + 180^\circ \\ \beta_{II} &= -(\beta_I + 180^\circ) \\ \gamma_{II} &= \gamma_I + 180^\circ \end{aligned} \tag{23}$$

Where α_i, β_i, and γ_i are rotations about the three axes with assignments between α_i, β_i, γ_i and $\boldsymbol{\theta_x}$, $\boldsymbol{\theta_y}$, $\boldsymbol{\theta_z}$ depending on the defined order of transformations.

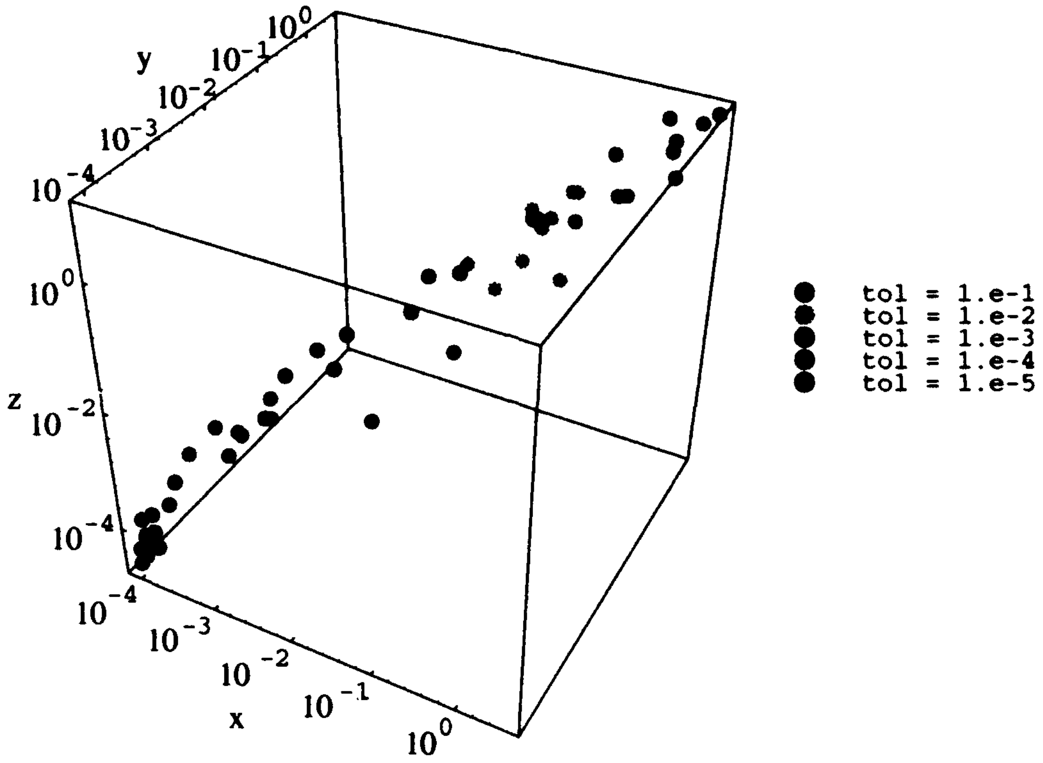


Figure 11. Logarithmic scatter plot of rms errors and convergence tolerance (inches).

These relationships, (23), are presented without rigorous proof, but can be understood intuitively. After the initial rotation, α, an object rotated in mode I is oriented in exactly the opposite direction of the object rotated in mode II. Rotation β results in a sort of mirror symmetry between the modes, and rotation γ brings both modes into the same orientation. These rotations are presented graphically in Figure 13.

Scale and translation errors are shown in Figures 14 through 16. Figure 14 shows some variation that increases with tolerance. The variation is, however, very small. Exact values of s can be found directly via (15) and (16), which do not require **mnewt** and are thus not affected by the tolerance setting. Similarly, the $x-$ and $y-$ translation errors (Figures 15 and 16) are not found via **mnewt** and do not vary with tolerance.

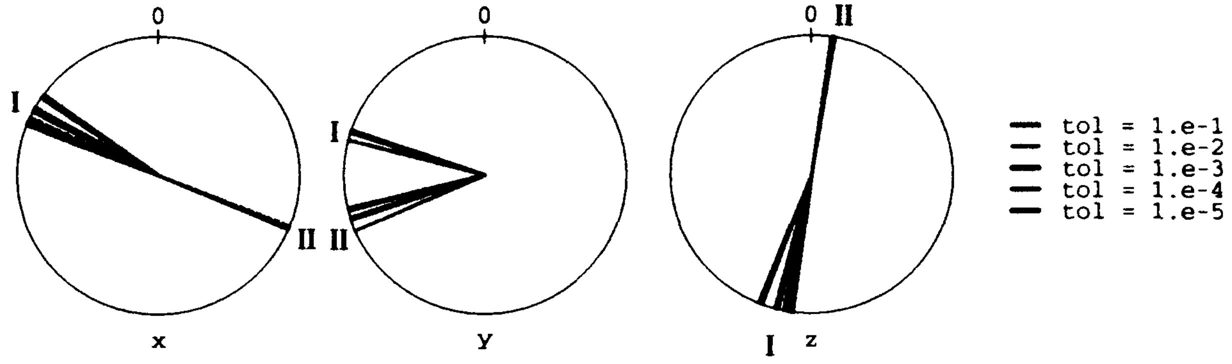


Figure 12. Rotational parameters and convergence tolerance.

4.2 Alignment Point Error

The location of alignment points has a marked effect on the quality of the extracted parameters. In the previous experiment, the alignment points were located at their true positions in the simulated sensor image. That is, no error was introduced in assigning alignment points. In this experiment, an error in the location of a single alignment point was introduced. It should be noted that as large errors in input alignment point locations are introduced, the convergence tolerance must be increased. The following experiments thus include the effects of both increased convergence tolerance and alignment error.

Three types of error were introduced. First, an error of fixed magnitude and direction was introduced, and the magnitude of the error varied. Second, an error of Gaussian distribution and variance based on the location of the alignment point was introduced to each coordinate of a single alignment point. This had the effect of varying magnitude and direction of the error. In the third case, the Gaussian errors were of an absolute, fixed variance. Each experiment is discussed in detail below.

4.2.1 Fixed Magnitude Error. For this experiment, errors, $\boldsymbol{\delta}$, were introduced to a single alignment point (Figure 17). The amount of error introduced was calculated by

$$\boldsymbol{\delta} = (cp_x, cp_y, cp_z)$$

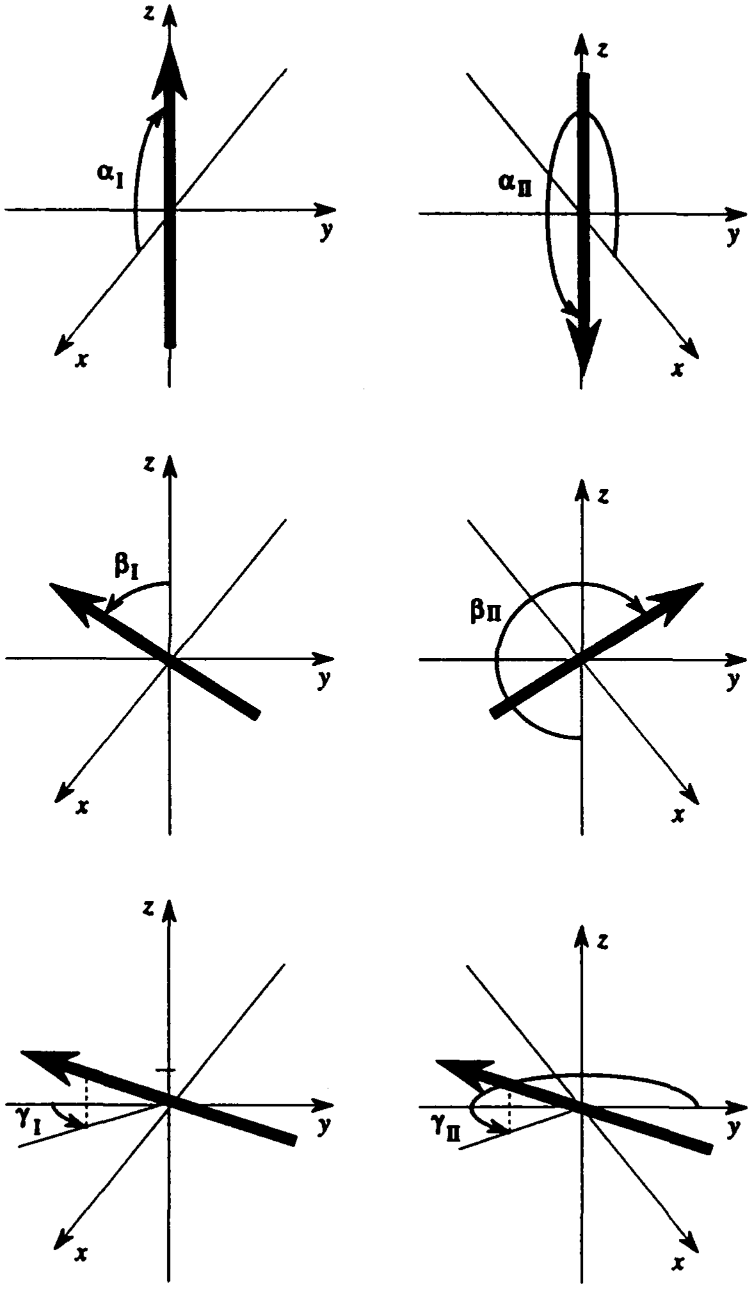


Figure 13. Two equivalent rotation modes.

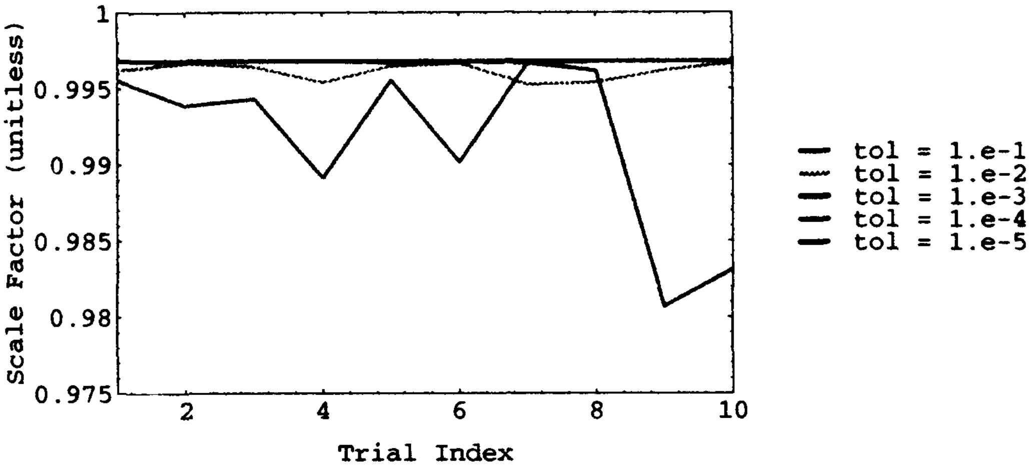


Figure 14. Scale parameter errors and tolerance.

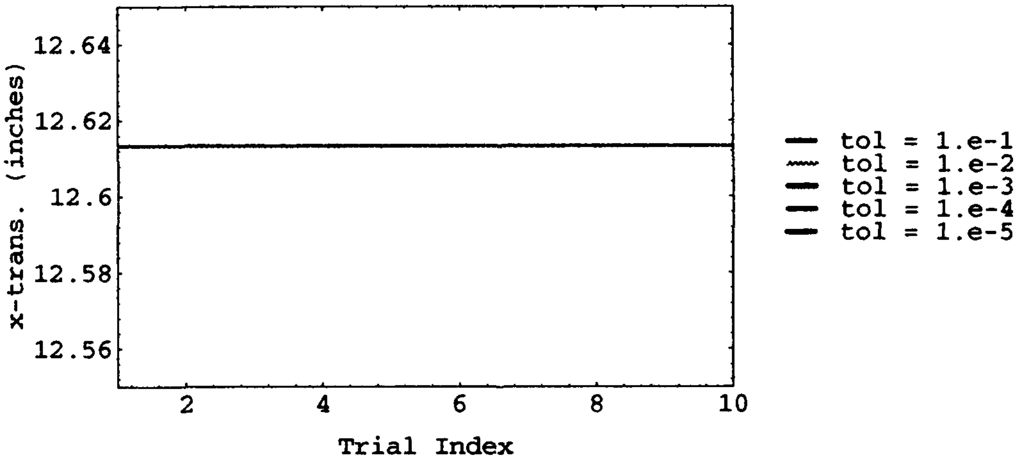


Figure 15. x translation parameter errors and tolerance.

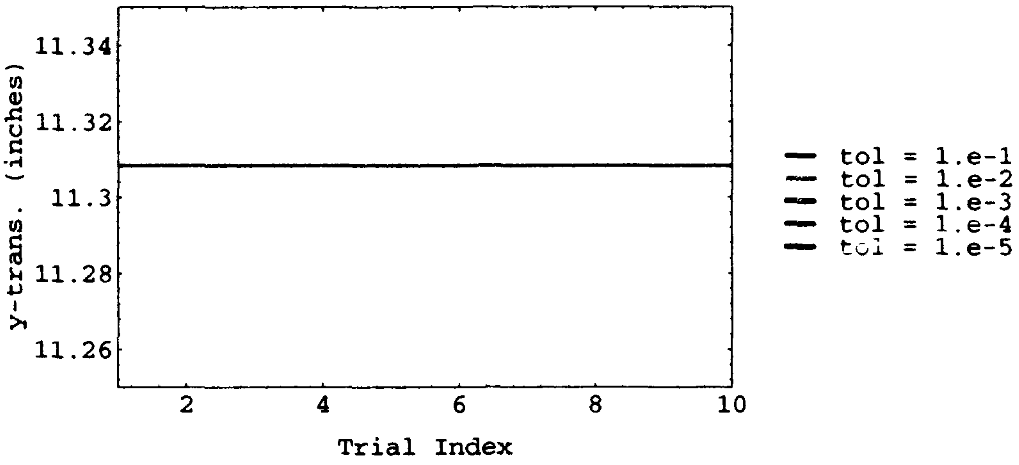


Figure 16. y translation parameter errors and tolerance.

where p_x, p_y, and p_z are the coordinates of alignment point $\boldsymbol{P}$, and c is the amount of error desired. The amount of error in each direction was varied from zero to 0.4, or 40%. The magnitude of the error was made relative to the coordinate magnitude because the coordinate magnitude is a rough measure of the object's size (assuming the object is centered near the origin, and the alignment point is chosen near the extremes of the object). With a relative error magnitude, generality and applicability to objects of any size are maintained. The same alignment point was moved for each trial, and ten trials were taken at each error setting. The tolerance was set to ensure convergence. Each trial was seeded with a randomly generated initial guess.

As in the previous experiment, each trial resulted in a parameter set $\boldsymbol{S}$ from which the following figures were generated. Figures 18, 19, and 20 indicate that both the magnitude and spread of output errors increased with the position error in the input alignment point. The translation error does not vary between trials since they are calculated explicitly.

4.2.2 Statistical Position Error. In the next two experiments, Gaussian errors were introduced into each coordinate of a single input alignment point of the satellite at an

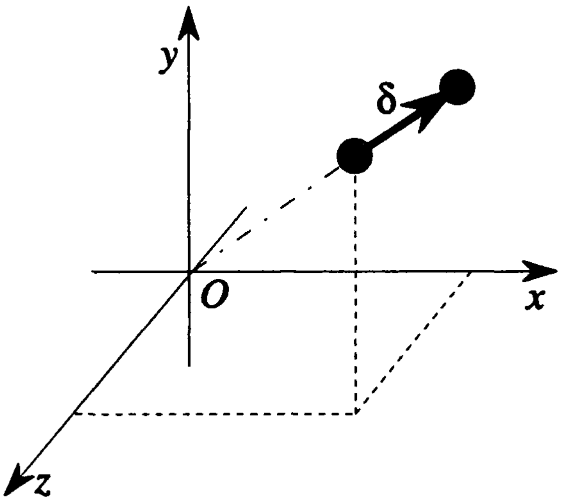


Figure 17. Introduced errors in alignment point.

arbitrary pose. These errors will be referred to as input errors. The same pose was used for each trial.

In one experiment, the variance of the error was based on the coordinates of the alignment point. In the other, all points were varied using a constant variance. Five hundred trials were performed at each of four alignment points, for both relative and absolute variance cases, for a total of 4000 trials.

In each case, the tolerance was set so that most of the trials converged. For each convergence, the resultant transformation parameters were applied to the reference model. The rms difference in the position of each alignment point in the transformed reference and the original pose were recorded as output errors. The results are presented below.

4.2.2.1 Relative Variance Error. For this experiment, a random error of Gaussian distribution and variance equal to 40% of the coordinate magnitude. The input alignment points were varied one at a time and a scatter plot (Figure 21) generated as described in Section 4.1. Figure 21 indicates that the output errors tended toward some

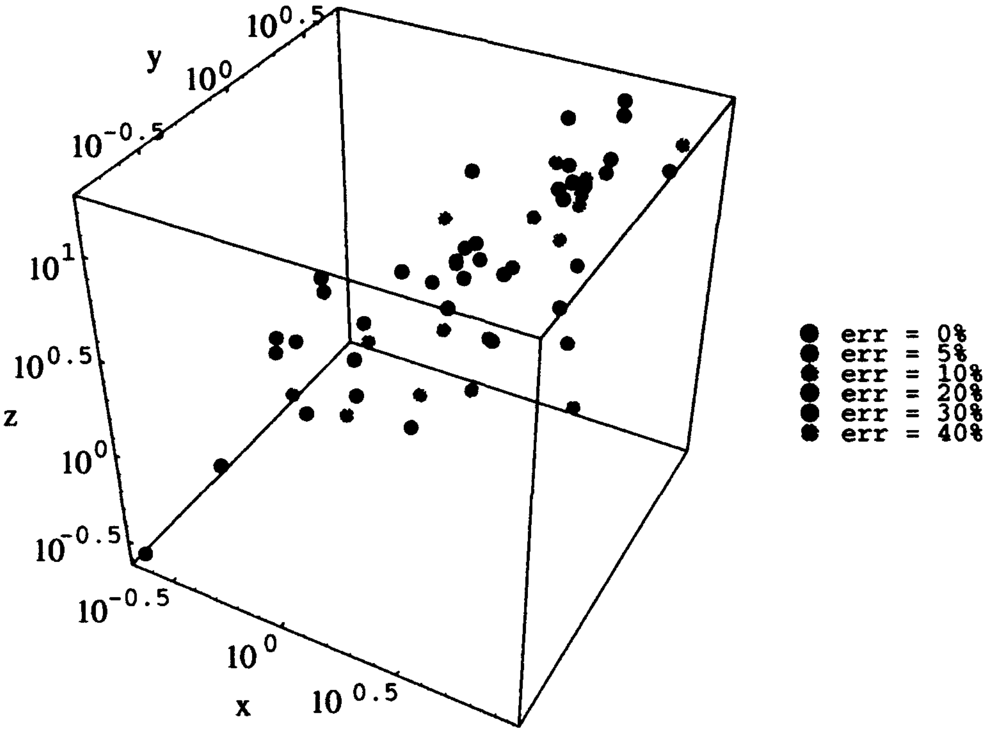


Figure 18. Logarithmic scatter plot of rms errors and alignment point error (inches).

median value, but with some very large errors. Linear plots and histograms of the rms output errors are shown in Figures 22 and 23.

Variance, median and mean of input and output errors are tabulated in Table 2. Note that the descriptive statistics of the output errors are calculated with respect to the rms values, and thus concern only the magnitude of the errors. The statistics on the input errors include both sign and magnitude.

The data suggest two possible patterns. First, the data indicate that the errors in z tend to be largest when compared to errors in x and y. Second, the data show that, the output errors have some dependency on which alignment point is varied.

The fact that the z errors are largest is probably a result of the fact that the data input to the system (the simulated sensor image) is of a two-dimensional nature. As a

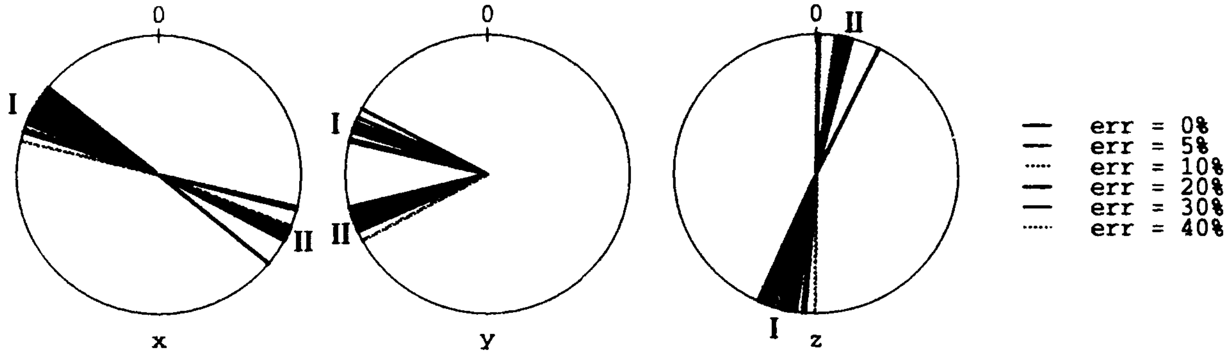


Figure 19. Rotational parameter error and alignment point error.

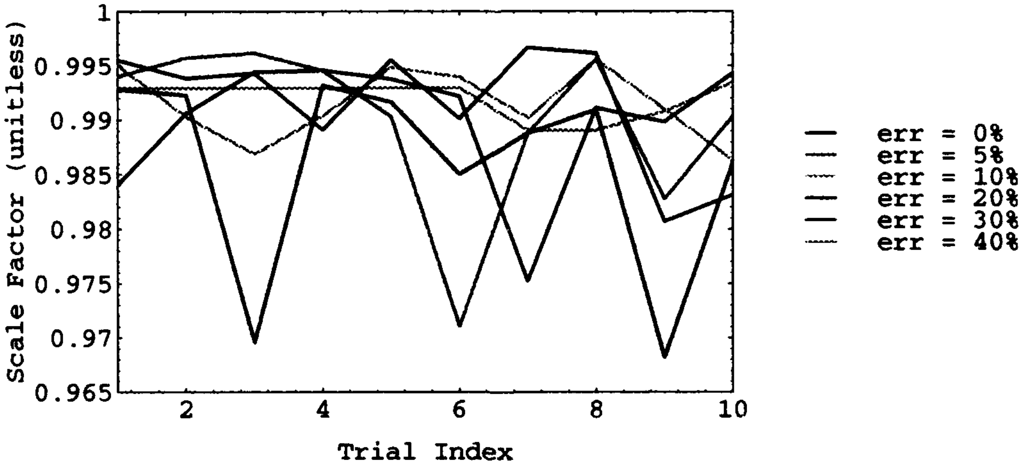


Figure 20. Scale parameter error and alignment point error.

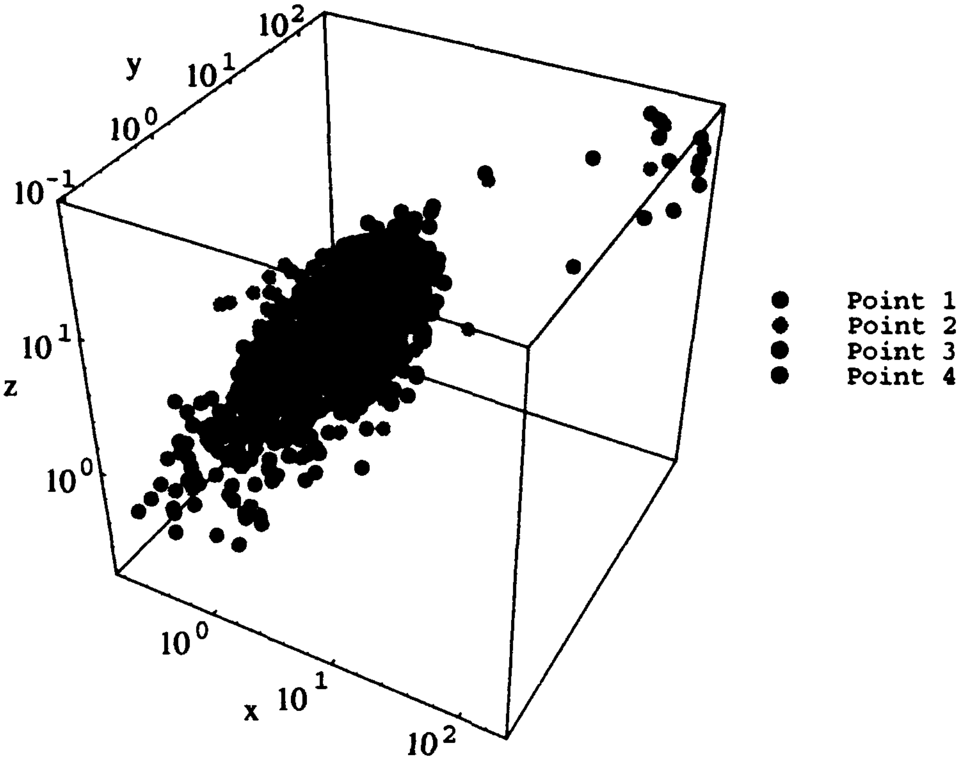
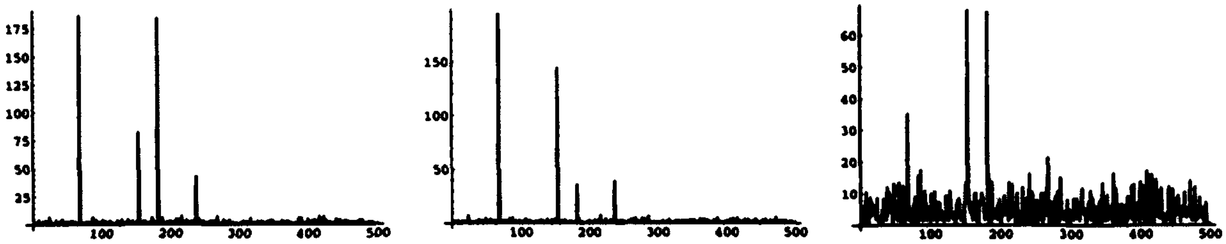


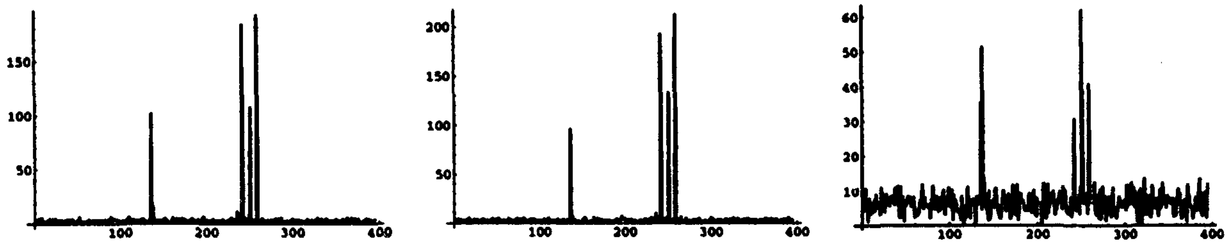
Figure 21. Scatter plot of output errors for relative variance error case. (inches)

Table 2. Descriptive statistics for relative error case.

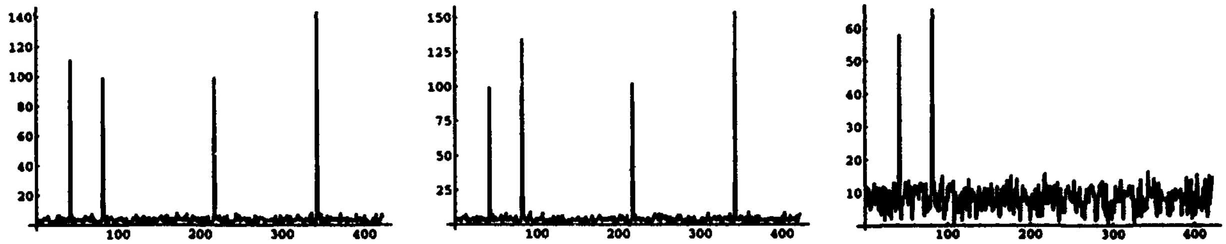
		Output (rms inches)			Input (inches)		
	AP No.	x	y	z	x	y	z
Mean	1	2.94208	2.55833	5.05387	0.0812185	-0.121072	0.249919
	2	4.44094	4.44663	6.8908	1.44183	0.100198	0.177558
	3	4.5709	4.87209	8.33287	-1.11337	1.27488	0.285058
	4	3.9526	4.34908	8.99865	-0.371578	-0.548609	-0.00138069
Variance	1	152.36	122.316	32.0131	9.81469	5.20771	25.7983
	2	226.408	268.213	27.0286	29.5943	27.5274	9.56482
	3	117.972	140.144	24.7852	43.1784	54.758	18.408
	4	149.245	173.624	28.5099	14.9638	38.6951	5.78199
Median	1	1.62021	1.45943	3.64331	0.214546	0.0122905	0.575096
	2	2.89291	2.818672	6.484473	1.270673	0.436209	0.174036
	3	3.38645	3.63636	8.29516	-1.16863	1.04323	0.137212
	4	2.980084	3.233486	8.530995	-0.338556	-0.528022	-0.047791



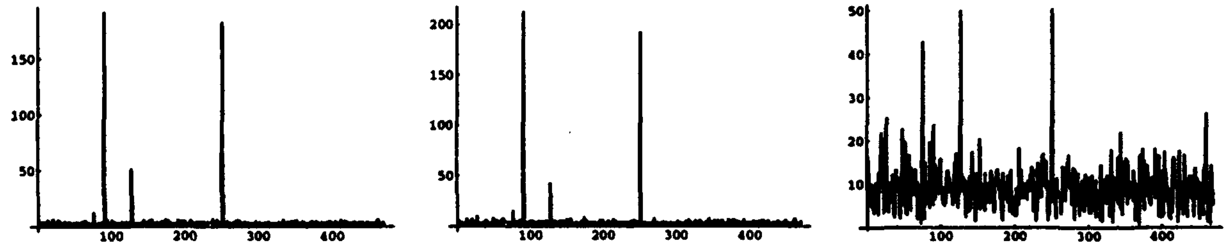
(a) x errors, AP 1 (b) y errors, AP 1 (c) z errors, AP 1



(d) x errors, AP 2 (e) y errors, AP 2 (f) z errors, AP 2

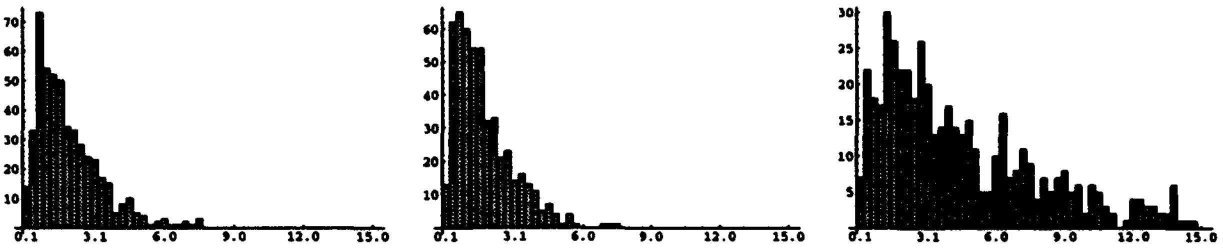


(g) x errors, AP 3 (h) y errors, AP 3 (i) z errors, AP 3

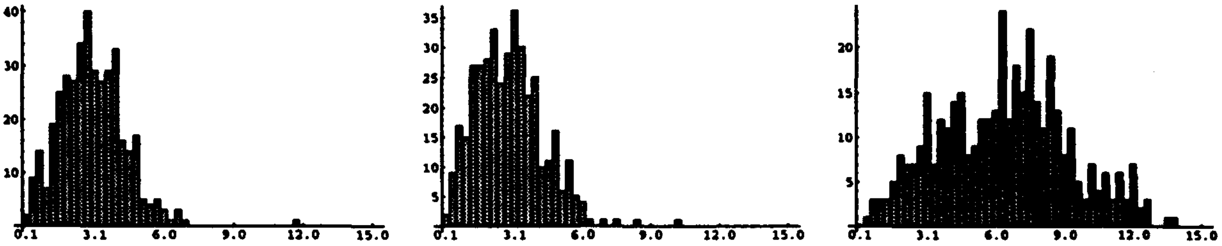


(j) x errors, AP 4 (k) y errors, AP 4 (l) z errors, AP 4

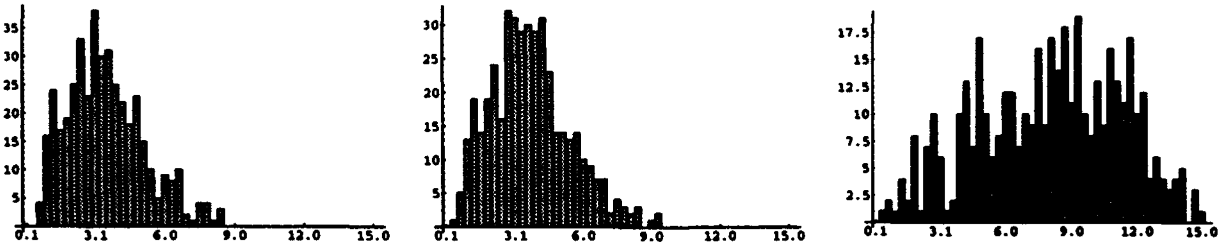
Figure 22. Rms error of all alignment points at output for relative variance error introduced at input of indicated alignment point. (rms inches vs. trial index)



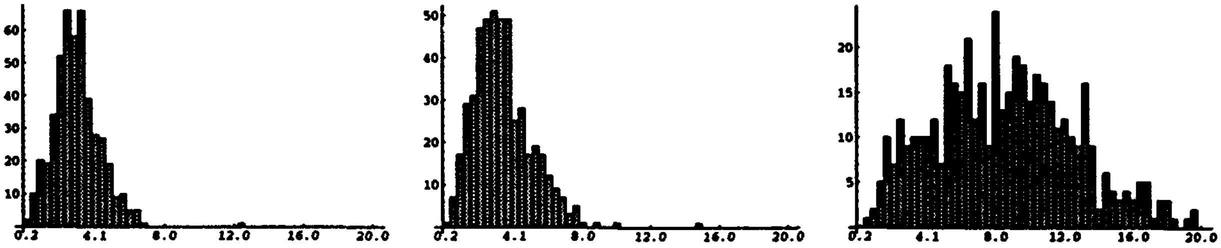
(a) x errors, AP 1 (b) y errors, AP 1 (c) z errors, AP 1



(d) x errors, AP 2 (e) y errors, AP 2 (f) z errors, AP 2



(g) x errors, AP 3 (h) y errors, AP 3 (i) z errors, AP 3



(j) x errors, AP 4 (k) y errors, AP 4 (l) z errors, AP 4

Figure 23. Histograms of rms output errors for all alignment points for relative variance error introduced at input of indicated alignment point. (rms inches)

Table 3. Descriptive statistics for absolute variance case.

		Output (rms inches)			Input (inches)		
	AP No.	x	y	z	x	y	z
Mean	1	4.28092	3.94261	8.9713	-1.38122	-0.261726	-0.148742
	2	4.86538	5.05233	7.36541	1.54298	-0.0219248	-0.692095
	3	4.11481	4.06205	8.44518	-0.783976	1.27228	-0.213669
	4	5.04234	4.97889	10.5366	-0.482599	-0.0133635	-0.0100668
Variance	1	157.244	180.993	61.628	51.7679	42.7191	64.6651
	2	309.401	345.356	42.3105	34.1778	39.4529	57.1569
	3	50.2505	59.5722	16.4593	53.4182	47.0717	56.7694
	4	150.672	161.375	52.2059	45.8241	48.4552	56.1919
Median	1	2.409773	2.206184	6.05825	-1.25679	-0.349007	-0.695875
	2	3.118494	3.131097	6.8591	1.205133	0.191433	-0.891528
	3	3.56763	3.48531	8.51234	-0.472554	1.21493	-0.459879
	4	3.75126	3.63944	9.74171	-0.782493	0.173531	-0.317917

result, no information on the z coordinates of the input alignment points is available. The only information available in the z dimension comes from the 3D model. Errors in the z dimensions are thus not observable to the system.

Because the alignment points are assigned arbitrarily, the output errors should not depend on which alignment point is in error. The relationship between the alignment point in error and the output errors are probably due to the differences in the variance of the input error. Recall that the variance of the input error in each dimension is a fraction (40%) of the true alignment point coordinate in that dimension. Thus, the input error is roughly correlated to the distance of the alignment point from the origin. The alignment points vary in distance from the origin, thus the input errors vary from alignment point to alignment point. The next experiment was conducted to test this theory.

4.2.2.2 Absolute Variance Error. To illustrate that the output errors are independent of which input alignment point is in error, the above experiment was rerun. An absolute variance of 60 inches^2 was introduced to each alignment point. Figures 24 through 26 and Table 3 summarize the results.

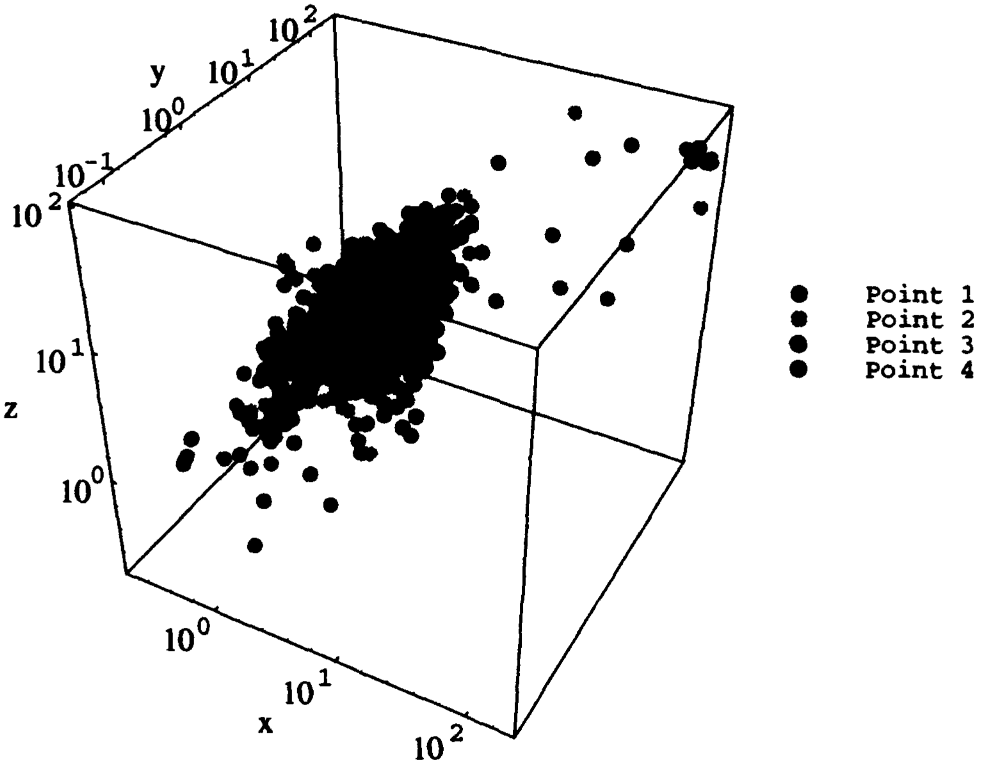
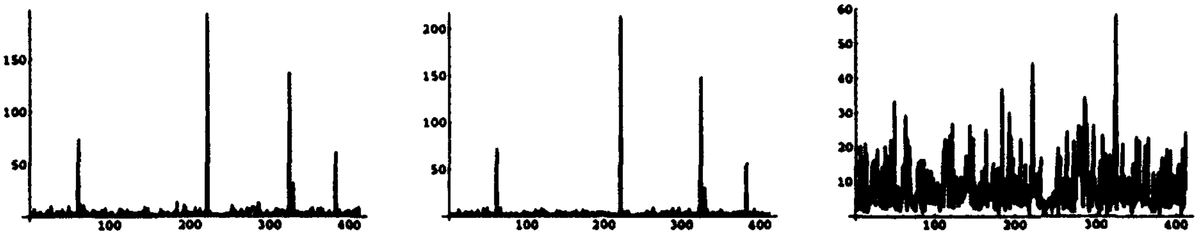
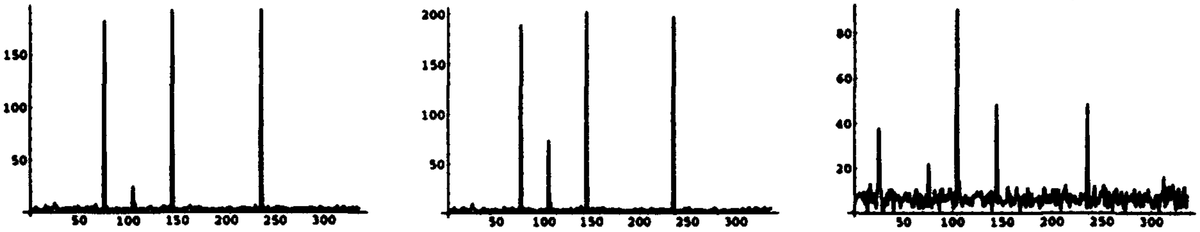


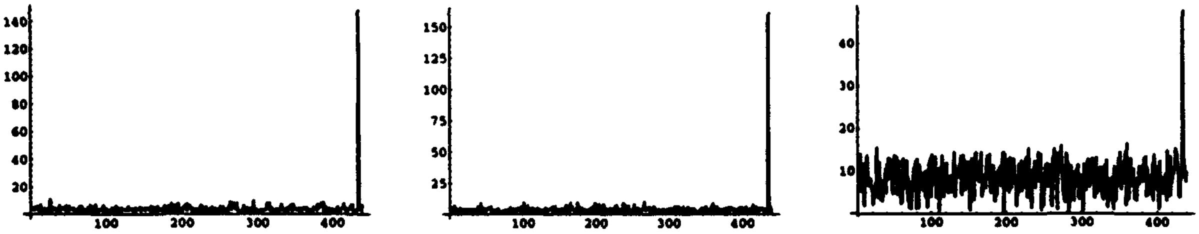
Figure 24. Logarithmic scatter plot of output errors for absolute variance case. (inches)



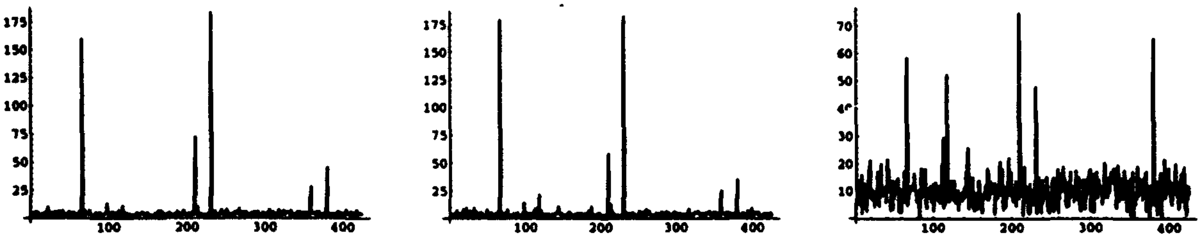
(a) x errors, AP 1 (b) y errors, AP 1 (c) z errors, AP 1



(d) x errors, AP 2 (e) y errors, AP 2 (f) z errors, AP 2

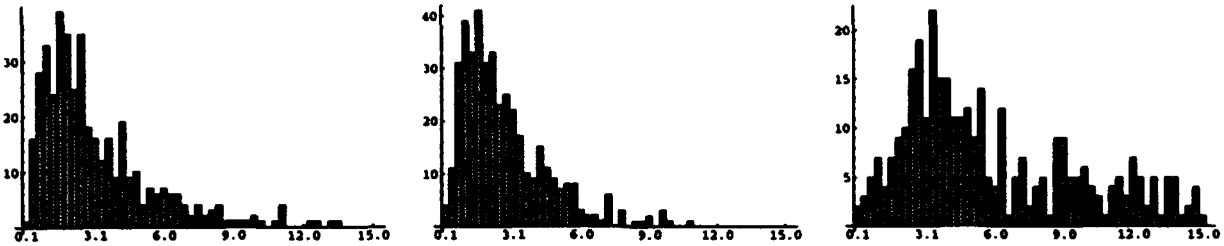


(g) x errors, AP 3 (h) y errors, AP 3 (i) z errors, AP 3

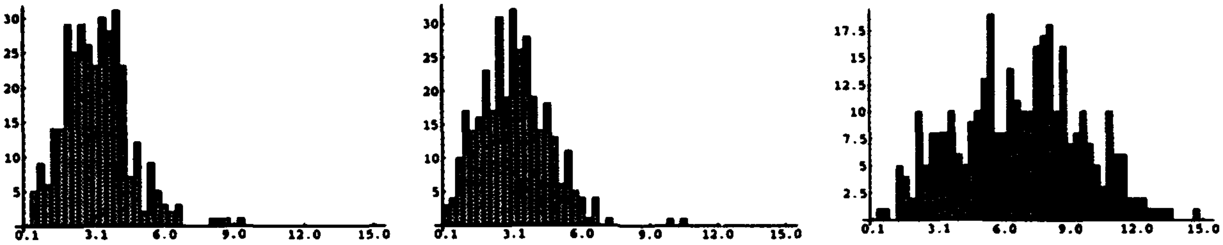


(j) x errors, AP 4 (k) y errors, AP 4 (l) z errors, AP 4

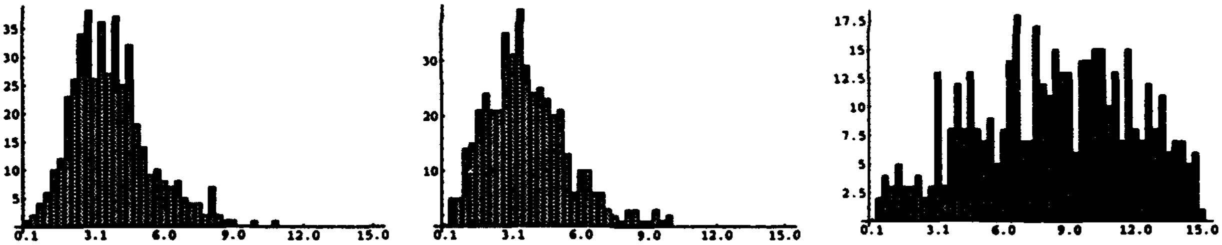
Figure 25. Rms error for all alignment points at output for absolute variance introduced at input of indicated alignment point . (inches vs. trial index)



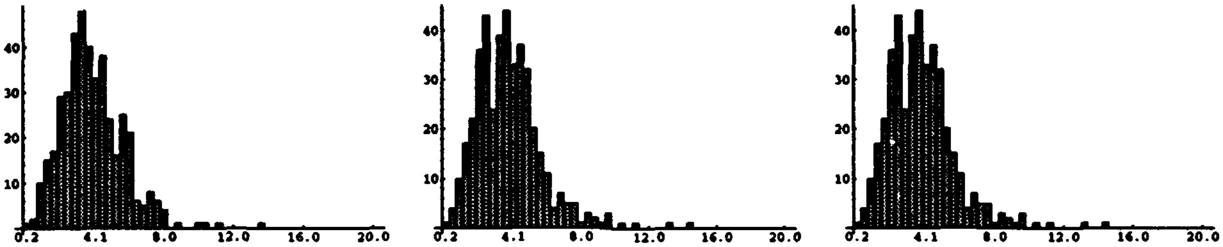
(a) x errors, AP 1 (b) y errors, AP 1 (c) z errors, AP 1



(d) x errors, AP 2 (e) y errors, AP 2 (f) z errors, AP 2



(g) x errors, AP 3 (h) y errors, AP 3 (i) z errors, AP 3



(j) x errors, AP 4 (k) y errors, AP 4 (l) z errors, AP 4

Figure 26. Rms error histogram for all alignment points at output for absolute variance error introduced at input of indicated alignment point . (rms inches)

The data indicate that some variation in the output errors still exists as the input alignment point in error is changed. Comparing the means and medians Tables 2 and 3 show that the variation is, however, much smaller for the case of absolute input error variance. The variation that does exist can be accounted for by the limited sample size. The variance measures in each case are very dependent on the few outputs with large errors. The fact that similar errors arise regardless of the alignment point that is in error indicates that output errors are not a function of which alignment point is in error.

4.3 Conclusion

The accuracy of the output of the attitude determination system is dependent on both the convergence tolerance and the errors in input alignment points. From this analysis of the performance of the parameter extraction system, a few general conclusions may be drawn.

A comparison of the two effects is difficult, since an error in input alignment point location requires relaxation of the convergence tolerance. Comparing the magnitude of errors as only the convergence tolerance is changed (Figure 11) with the magnitude errors as errors are introduced (Figure 17) indicates that output errors increase with tolerance, and high tolerances contribute a great deal to output errors.

Input alignment point errors require larger convergence tolerances. Thus the true effect of input alignment point errors may be somewhat masked by the effect of the high tolerance required for convergence, but input alignment point errors undoubtedly contribute to output errors.

Errors in the output of the alignment parameters can be observed in many representations. The actual parameters (Figures 12, 14, 15, and 16, for example), or the rms errors resulting from transformation by the parameters (*e.g.* Figure 11) can be studied. Inspection of the actual parameters shows that the rotational parameters occur in two modes that both give equally valid results. Rms errors provide a quantifiable measure allowing direct comparison between solutions and calculation of statistical measures. Examination of rms errors shows that relatively large errors in the z dimension occur, but are generally not observable in a 2D image.

Errors in output vary according to the magnitude of the location error at each input alignment point, and not according to which alignment point is in error. That is, equal errors among alignment points contribute equally to output alignment errors. Because output errors arise through a non-linear mapping from input errors, superposition of output errors will not hold. That is, output errors when four alignment points are in error will not be the sum of the errors introduced by the error in each alignment point. The case of multiple alignment point errors was not studied, and further investigation is necessary before drawing conclusions other than the above.

Because vector images were used in the above analysis, fine variations in alignment point locations and errors could be introduced. In a more realistic application, pixel images would be used, and precision in identification of alignment points in the image cannot exceed the pixel resolution of the image. The error statistics used in this analysis are analogous to a single pixel error at a hypothetical resolution of one foot square.

In studying the error sources and mechanisms discussed in this chapter, it should be noted that they are specific characteristics of the extraction method used. In this case they are characteristic of the Newton-Raphson method for extraction of scale and rotation, and the linear algebra expressed by (22). Implementation of alternate methods would require additional analysis of their performance before conclusions can be drawn.

V. Conclusions and Recommendations

The principal goal of this thesis was to develop a technique for determining the attitude of an object, specifically a satellite, in a 2D image. The technique developed estimates the alignment parameters that transform a 3D model from a reference attitude to an attitude that can be projected to match the attitude in the image. Additionally, coefficients are found that can combine a series of basis images to form an image of the object at the same pose as in the original image.

The basic techniques of the alignment parameter and coefficient extraction were initially modeled in *Mathematica*, then implemented on a Silicon Graphics workstation using C and the SGI Graphics Library (GL). The techniques were tested on a simple 3D figure and a satellite model obtained from the Philips Laboratory Satellite Assessment Center.

A performance analysis was performed on the SGI implementation. Qualitative effects of input alignment point errors and tolerance parameters in the numerical method employed were observed and described. As expected, output errors increase with convergence tolerance and input alignment point errors.

This chapter covers the basic findings of this thesis, and outlines areas for further investigation.

5.1 Research Conclusions

A 2D image of a 3D object at an arbitrary pose can be generated as the linear combination of three basis images. Embedded in the coefficients of the linear combination is information specifying the pose of the object in the original image. That information can be successfully extracted. Using the techniques developed in this thesis, the amount of data stored beforehand can be reduced to a minimum of three 2D basis images of the object of interest at known poses.

With a system such as the one developed for this thesis, it is possible to estimate the pose of a 3D object represented in a 2D image. To do so, one must be able to associate four alignment points in a set of 2D basis images with their locations in the original 2D image. A set of coefficients that combine the basis images to form the original image and specify the translation parameters can then be found. Combining the coefficients with a 3D model of the object or pose information from the basis images results in a system of non-linear equations. The equations can be solved numerically to yield a set of transformation parameters that describe the pose of the object in the 2D image.

The quality of the estimated transformation parameters depends on the method used to solve the non-linear system of equations and the accuracy with which the alignment points can be associated between the images and the model.

The main parameter in the *Numerical Recipes* Newton-Raphson method used in this thesis was the convergence tolerance. The use of high tolerance settings resulted in degradation in the quality of the extracted parameters as measured by the rms error between the true alignment point locations and the locations derived from application of the extracted alignment parameters. High tolerance settings are sometimes required if errors in location of the input alignment points exist.

Errors in the location of input alignment points also degrade the quality of the extracted transformation parameters. Errors in output, evident in both the rms measure described above and comparison of the alignment parameters with the true values, increase as errors in the input alignment points increase. Output errors are not a function of which alignment point is in error.

5.2 Contributions

The main contributions of this thesis are twofold. First, it combines the principle of linear combination of images and the concept of homogeneous coordinate systems. This allows all transformations of interest (scaling, rotation, and translation) to be modeled as linear transformations, which in turn allows calculation the coefficients for linear combination of the basis images in a single matrix multiplication. Additionally, a scheme for choosing basis images so that translation parameters are available by inspection from the coefficients is described. A method is also described for determining the other transformation parameters from the linear combination coefficients.

The second contribution is the construction of a software tool that applies the techniques developed above. The tool generates vector images from a constructive solid geometry (CSG) model of a satellite that can be used as basis images or as simulated sensor images. From these images, the tool calculates the linear combination coefficients and can make reasonable estimates of the transformation parameters used in posing the satellite for the simulated sensor image.

5.3 Recommendations

This thesis has illustrated the basic utility of the parameter extraction method studied. The system developed here represents a single implementation using only a few of the many possible implementation techniques. Several alternate techniques exist. A few were alluded to earlier, but not covered in detail due to the limited scope of a thesis effort. These techniques may provide better performance and are worthy of further investigation. Additionally, opportunities exist for further study and improvement of the implementation described here. In many of these areas, the potential exists for eliminating the need for the 3D model, thus reducing the requirement for reference data to the basis images.

5.3.1 Alternative Parameter Extraction Techniques.

5.3.1.1 Analytical Methods. Alternative perspectives of the problem of parameter extraction suggest that analytical (not iterative and non-numerical) techniques may exist for extraction.

The linear combination coefficients, $\boldsymbol{a}$ and $\boldsymbol{b}$, and the columns of the composite transformation matrix, $\boldsymbol{T}_c$ can be viewed as orthogonal vectors. Each set of vectors can be normalized and taken to represent the axes of transformed coordinate space. The transformation between the coordinate spaces may be somehow analogous to the transformation of the object, thus yielding the object transformation parameters analytically.

5.3.1.2 Non-Linear Methods. In the current implementation the required elements of the composite transformation matrix cannot be found analytically without the 3D geometric model. Information about the transformation parameters, along with the 3D geometry, are embedded in the linear combination coefficients.

The linear combination coefficients can be found analytically. With the choice of basis images outlined above, the translation parameters are determined in the process of calculating the coefficients. This leaves four parameters, scaling and rotation about each axis, that must be found through additional computation.

Further study of the coefficients and the alignment parameters may reveal a relationship that may be modeled by a non-linear technique. It is possible that neural networks or hyperbasis functions (basis functions in more than three dimensions (12)) may be employed to map the linear combination coefficients directly to transformation parameters.

5.3.2 Potential Improvements. The current system is a rudimentary implementation of only the basic elements of the parameter extraction methods. Improvements are possible that can increase the functionality and performance of the system. A few of these improvements, which fall roughly in the area of artificial intelligence (AI), are discussed below (15).

The system relies on a series of alignment points. Corresponding points in the 3D model, basis images, and the sensor image must be identified. The alignment points must meet certain criteria that describe their spatial relationships to each other in 3D. Additionally, the alignment points should be at the extremities of the object so that they can be identified easily in the sensor image. Currently, the alignment points are chosen manually in the 3D model. An AI system could be built to automatically choose alignment points based on the criteria described above.

Once the alignment points are determined and a sensor image obtained, the user manually identifies the alignment points in the sensor image. Sometimes, one or more alignment points may be obscured due to the pose of the object. Additionally, the sensor images may be noisy, making it difficult to identify the alignment point exactly. An AI system could be useful at several levels: it could help the user to locate the obscured alignment points; it may choose and perform image enhancement processes to reduce noise, facilitating precise location of the alignment points; it may locate the points precisely given a rough estimate from the user; or it may locate all the alignment points automatically.

5.4 Summary

This thesis has shown that it is possible to extract information describing the pose of a 3D dimensional object in a 2D image. While this study focused specifically on the problem of satellite attitude determination, the methods discussed are applicable to any object on which suitable alignment points may be defined.

Appendix A. User's Guide

This appendix contains detailed information on how to use the software tools that make up the attitude determination system implemented for this thesis. Three main tools are included: `writesat` for model manipulation and image generation; `matcher` for attitude determination and coefficient extraction; and `sens` for performance analysis of the numerical routines used. A listing of all source code is included in a subsequent appendix.

The system estimates the attitude of a satellite in a 2D image by extracting the parameters that will transform a 3D model of the satellite from an arbitrary position to one that will produce the 2D image when projected into the xy plane. The attitude is thus relative to the original, arbitrary position.

The main component of the system is the `matcher` program, which performs the calculations necessary to extract the transformation parameters from the input data. The `writesat` program is a utility that manipulates the 3D model of the satellite of interest, and generates output files that are used as input images and data for `matcher`. The utility, `sens`, is an associated program for performance analysis through repeated executions of the numerical routines used by `matcher`.

The programs were written and are run on a Silicon Graphics Workstation in C using the Silicon Graphics Graphics Library (GL) and numerical routines from *Numerical Recipes in C.*

A.1 Model Manipulation

This program, `writesat` generates the input images for `matcher`. Required inputs are a constructive solid geometry model (CSG) file of the satellite of interest, and a control file containing input and output filenames and transformation parameters which specify a pose.

A.1.1 Input File Formats.

A.1.1.1 Control File. The program is controlled using an input file provided from the UNIX command line.

```
C1: Control file for writesat
C2: Martin Chin
C3:
C4:
In: ../satmodel/dmsp.geom.nsm.alp
Out: bas1.vec
1.0 60.0 0.0 0.0 0.0 0.0
```

The control file, listed above, must contain the filename of satellite model, the desired pose (s, θ_x, θ_y, θ_z, t_x, t_y)of the satellite in the output image, and an output filename. If a scale factor of zero is indicated by the control file, a random pose is generated.

A.1.1.2 3D Model File. The 3D model is stored in a modified New Solid Model (NSM) format. The NSM format is generated by the SMT satellite modeling program used by the Philips Laboratory Satellite Assessment Center. The data is stored in a hierarchical structure. Simple assemblies, or model elements, are composed of eleven primitives including rectangular parallelepipeds, boxes, wedges, hexahedrons, cones, elliptical cylinders, elliptical cones, spheres, hemispheres and ellipsoids. More complex assemblies are then composed of the model elements or other assemblies. Finally the entire satellite is represented by a few rather complex assemblies. (2) NSM format files are customarily given the `.nsm` suffix. The modified format is distinguished by the suffix `.nsm.alp`.

The NSM file format includes four main sections: Header, Model Element Definitions, Model Assembly, and Final Assembly. The sections must appear in the order listed above.

The header contains generic information about the model, such as its name, units used in the file, and the number of parts included in the file. The header is limited to the first three lines of the file.

```
DMSP Block 5D-2
UNITS=IN
LAST UNIQUE ID = 1582
```

The satellite model used in this thesis is the Block 5D-2 DMSP, supplied by the Philips Laboratory Satellite Assessment Center.

The model element definitions section lists information on the composition of model elements and their positions.

```
#
# Model Elements
#
$
$  Alignment Points
$
AP 0 65.379997 13.509999 -21.000000
AP 1 -25.040001 127.193779 114.209000
AP 2 -12.882499 -156.188995 119.519005
AP 3 0.000000 1.194350 -127.111397
$
SAVE  Apogee Kick Motor                     EL   2
#Surface Property: Ti-6Al-4V titanium alloy (bare)
#Bulk Property: Ti-6Al-4V titanium
#Minimum Thickness:      0.752
#Initial Temperature (Degrees Kelvin):       298
#Melt Removal Flag: Melt Temperature Plus Phase Change
#Damage Mode #1: NONE                                   # of Rays: 0
#Weibul Parameters: A:         1 B:           1 C:           1
#Damage Mode #2: NONE                                   # of Rays: 0
#Weibul Parameters: A:         1 B:           1 C:           1
#Damage Mode #3: NONE                                   # of Rays: 0
#Weibul Parameters: A:         1 B:           1 C:           1
$
$
$
S SPH            0.0        0.0        0.0       0.0X       0.0Y       0.0Z
         0          0          0       16.5
#External: YES
#Collector Points:  NO   Spacing:      5   Minimum #:       1
#Faces =  1: NO     2: NO     3: NO     4: NO     5: NO     6:  NO
V SPH            0.0        0.0        0.0       0.0X       0.0Y       0.0Z
         0          0          0     15.748
#External:  NO
#Collector Points:  NO   Spacing:      5   Minimum #:       1
#Faces =  1: NO     2: NO     3: NO     4: NO     5: NO     6:  NO
```

```
END   Apogee Kick Motor
#
```

It must begin with the first three lines displayed above. Each element is then delineated by the **SAVE** and **END** keywords. Each element is given a unique name and integer identification number. Between the keywords is various information about the construction of the element denoted by a # at the beginning of the line. Comment lines are denoted by $. Lines beginning with # or $ are usually ignored for the purposes of this thesis.

Modifications to the file are limited to location of alignment points in the file. Alignment point designations must be embedded in the Model Elements section of the NSM file and referenced to the same coordinate system as that of the final assembly. The new keyword **AP** is used to denote alignment points. Each alignment point record begins with **AP** an includes the number of the alignment point and its location.

A list of primitives with appropriate parameters for size and location then follows. Each primitive is denoted by an **S** or **V**, indicating whether the primitive is solid or a void, followed by a three letter code indicating the type of primitive. Only solid the primitives are of interest in this thesis.

The Model Assembly section describes the structure and position of the assemblies.

```
#
# Model Assemblies
#
SAVE  prop.L3.N2H4 Thruster                CM   526
CALL  N2H4 Thruster                                                  527
        0         0         0        0X        0Y        0Z
EXTERNAL: YES
END   prop.L3.N2H4 Thruster
#
```

Each assembly is also delineated by **SAVE** and **END** keywords. Each assembly is named and given a two letter code indicating the type of the assembly (**CM** = component, **SF** = subfunction, **SS** = subsystem, **SY** = system). The **CALL** keyword then refers to name of the model element or assembly to be used, and provides transformation parameters for its location in the assembly.

The final assembly section calls the assembly that describes the entire satellite. Note that no `SAVE` and `END` keywords are used.

```
#
# Final Assembly
#
CALL  SYSTEM DMSP
        0.0       0.0       0.0       0.0X      0.0Y      0.0Z
#
LAST
```

The end of the file is signaled by the keyword `LAST` on the last line of the file.

A.1.2 Execution. The program is started from a C shell command line by the following command at the UNIX prompt:

`writesat` *filename*

where *filename* is a control file of the format specified above. `writesat` then opens the required files and displays the model at the specified pose.

The alignment points are displayed as colored spheres (Figure 9), so that they are visible at all poses. The user may then manipulate the rotation and scaling of the pose before generating an output file. The user's focus (the SGI cursor) must be in the `writesat` window to control the program.

A.1.2.1 Pose Manipulation. To change the pose, the numeric keypad is used. The numerals must be enabled by pressing the Num Lock key. An indicator light on the keyboard denotes the Num Lock mode. Figure 27 illustrates the rotations invoked by the keys. The 5 key restores the original pose specified in the file or the originally generated random pose. The + and - keys change the scale factor up and down, respectively. The new transformation parameters are displayed in the C shell window.

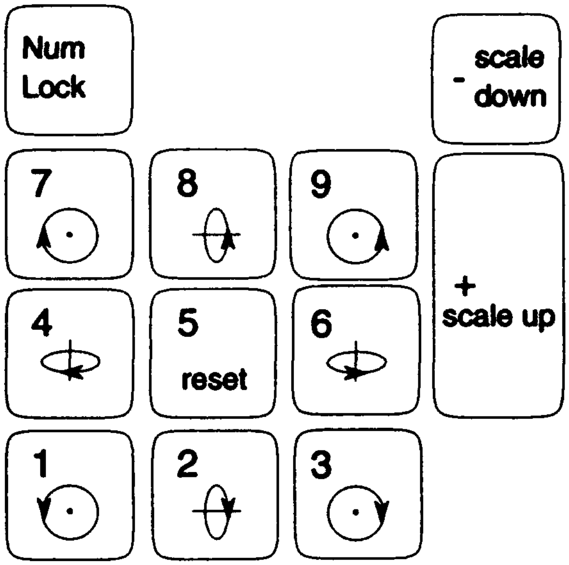


Figure 27. Numeric keypad functions in for pose manipulation.

A.1.2.2 Output File Creation. When a suitable pose is reached, the output file with the name as specified in the control file is created by pressing the space bar. The file is generated in a vector format. The transformation data is stored in the header, followed by 3D alignment point locations, then the 3D vector endpoint pairs. A segment of the output file illustrating the format follows:

```
Model: ../satmodel/dmsp.geom.nsm.alp

Scale: 0.996811
   Rx: 67.381210
   Ry: 108.035217
   Rz: 170.946991
   Tx: 12.613300
   Ty: 11.308329

AP: 0 24.570559 -15.408206 -63.325493
AP: 1 -137.169266 92.209892 -26.056511
AP: 2 127.339104 165.023773 42.519070
AP: 3 38.865135 -111.773666 14.746902

-4.198954 26.277088 -23.518766 -5.126214 25.999113 -24.111668
-5.126214 25.999113 -24.111668 -5.625136 25.766003 -25.104317
```

```
-5.625136 25.766003 -25.104317 -5.562035 25.640219 -26.230732
-5.562035 25.640219 -26.230732 -4.953817 25.655464 -27.189091
```

The output files for complicated objects like the DMSP satellite are rather large and require several seconds to write. The program may be exited by choosing Close or Quit under the `writesat` window menu.

A.2 Alignment Parameter Extraction

The `matcher` program is the main element of the parameter extraction system. It performs all calculations for determining the alignment parameters that specify the pose of the satellite. It requires several input files: a control file, specifying the file names of the required files; a reference file specifying the reference pose of the satellite; three basis images, for which linear combination coefficients will be generated; and a simulated sensor image, for which the pose of the satellite will be determined. An optional model file may be supplied to provide a 3D model on which the extracted parameters may be tested.

A.2.1 Input File Formats.

A.2.1.1 Control File Format. The control specifies the filenames for all the input files.

```
C1: satmaster.in
C2:
C3: Master control file for matcher program.
C4:
RE: ref.vec
B1: bas1.vec
B2: bas2.vec
B3: bas3.vec
UM: rand.vec
MD: ../satmodel/dmsp.geom.nsm.alp
```

A.2.1.2 Image File Format. The image files are generated by the `writesat` program, and are of the format described above. In total, five images (1 reference, 3 basis, 1 simulated sensor) are required. Thus `writesat` must be run five times to supply the required data files for the initial execution of `matcher`. Once the basis and reference image files are generated for a particular satellite, they may be used repeatedly with different simulated sensor images.

As noted above, `writesat`'s output files are actually 3D vector files. The files specified as images are automatically projected to an orthographic view as they are read in, converting them to true 2D images.

A.2.1.3 Model File Format. The model file format is the same as that used by `writesat`, and is covered in detail above. This file is optional for execution of `matcher`, but its inclusion is recommended so that the quality of the extracted parameters can be gauged. The model is transformed by the extracted parameters, and should match the simulated sensor image once it is transformed.

A.2.2 Execution. The program is launched from a C shell command line using the following syntax:

```
matcher filename [-ssize] [-upixels]
```

where *filename* is the control file and the follow command line options are available.

`-s`*size*	World coordinate dimensions of matcher window
`-u`*pixels*	Screen coordinate dimensions of matcher window

The program opens windows to display the basis images, simulated sensor image, and 3D model (if supplied). A picture of the SGI screen is provided as Figure 10.

The program provides a pop up menu in the main `matcher` window. To display the menu, the menu (right) button is pressed while the focus is in the main `matcher` window. Each selection is described below:

A.2.2.1 Calculate Coefficients. Selecting this menu item causes `matcher` to calculate the coefficients for the linear combination of the basis images that will form the simulated sensor image. This item is disabled until all the alignment points have been identified in the sensor image. The coefficients are output to the C shell window from which `matcher` was launched.

A.2.2.2 Calculate Parameters. Selecting this menu item causes `matcher` to calculate the alignment parameters that describe the pose of the object in the simulated sensor image. This item is disabled until all of the alignment points have been set in the simulated sensor image.

If the parameters are found, they are printed to the C shell window from which `matcher` was launched. If no solution is found, an error message is printed. In some cases, a solution can be found by simply selecting Calculate Parameters again. Usually, the failure to converge to a parameter set results from inaccurate placement of alignment points in the sensor image. The alignment points can sometimes be placed more accurately using the Zoom features described below. If the alignment points cannot be placed with sufficient accuracy, then the convergence tolerances may be adjusted.

A.2.2.3 Zoom. The Zoom feature allows the user to enlarge a section of the simulated sensor image. The user simply selects the portion of the sensor image to be enlarged by placing a zoom box on the image, then selecting the Zoom In item from the Zoom sub menu. The zoom box is placed by pressing and holding the left mouse button at one corner of the area to be enlarged. A magenta cross is displayed to mark the spot where the button was pressed. Holding the button down, the user drags the cursor across the area to be enlarged and releasing the button at the diagonally opposite corner of the area. A magenta zoom box then appears on the image. To prevent warping in the zoomed image, the zoom box is automatically squared so that it encompasses the entire area selected by the user.

Selecting Zoom In from the Zoom sub menu enlarges the area surrounded by the zoom box. The Zoom In item is disabled unless a zoom box has been set by the user.

Selecting Unzoom returns the original view of the entire image.

A.2.2.4 Set Alignment Points. The Set Alignment Points feature allows the user to place the alignment points in the simulated sensor image. Selecting Set Alignment Points causes the display of a submenu which allows the user to set individual points, set the points to predetermined locations, or clear all the points. The location of a particular point can be modified by simply choosing to set that point again. When used in combination with the Zoom feature, the alignment points can be set very accurately.

Red, Green, Yellow, Blue

Selecting Red, Green, Yellow, or Blue from the Set Alignment Points sub menu allows the user to set or reset the red, green, yellow, or blue alignment point, respectively. After choosing the point to be set, the user simply click the left button of the mouse when the tip of the cursor is at the desired location for the alignment point. If the alignment point is already set, it is reset to the new location. The old location is displayed until the mouse is clicked.

Auto

This feature sets all points simultaneously to their true locations as specified in the image file header. Any points which were previously set are reset to the locations specified in the file header.

Clear

Selecting Clear from the Set Alignment Points sub menu clears any alignment points that may have been set.

A.2.2.5 Tolerance. This feature allows the user to change the convergence tolerance used in the extraction of the alignment parameters. Selecting Tolerance displays a sub menu from which Increase or Decrease may be selected. The tolerance (`tolf` and `tolx` as described in 4.1) is adjusted up or down by factors of 10. The new setting is displayed in the C shell window from which `matcher` was launched.

A.2.2.6 Quit. Selecting Quit exits the `matcher` program.

A.3 Performance Analysis

This program, **sens** calls the numerical routines used by `matcher` to extract alignment parameters. Several options allow the variation of extraction parameters such as convergence tolerance and alignment point error. It processes a fixed set of data (Table 1) so no input files are required. It outputs a file containing the extracted parameters, any errors introduced, and rms output errors.

A.3.1 Output File Format. The output file contains a record for each execution of the alignment extraction routine. Each record contains the extracted parameter set (s, θ_x, θ_y, θ_z, t_x, and t_y), rms errors (x, y, and z), and alignment point errors (x, y, and z), giving a total of twelve elements.

Rms errors represent the errors between the true positions of the alignments points and the locations of the alignment points after transformation of the model by the extracted alignment parameter set.

A.3.2 Execution. The program is executed from a C shell window using the following syntax:

sens *outfile* [-t*tol*] [-l*loops*] [-e*err*] [-p*varpt*]

The command line options available are as follows:

-t*tol*	Sets convergence tolerance to *tol* Default value = .00001
-l*loops*	Sets the number of loops to *loops* Default value = 10
-e*err*	Sets introduced error variance to *err* inches Default value = 0
-p*varpt*	Varies alignment point *varpt* $\in 1, 2, 3, 4$ Default value = 1

If a command line option is not exercised, the default value is used.

Errors are generated at random with a Gaussian distribution.

Appendix B. Mathematica Source Code

The following is a listing of the *Mathematica* code used in modeling the coefficient and parameter extraction techniques.

```
four:={{-2,2,2,1},{2,2,2,1},{0,-2,2,1},{0,0,-2,1}}

pt[list_,i_]:=Take[list[[i]],3]

rem[num_]:=Mod[N[Abs[num]],N[2*Pi]]*Sign[N[num]]

obj[list_]:=Graphics3D[{{AbsoluteThickness[4],
Line[{pt[list,1],pt[list,2],
        pt[list,3],pt[list,4]}]},
        {PointSize[0.05],GrayLevel[0.5],Pcint[pt[list,1]]},
        {PointSize[0.05],Point[pt[list,2]]},
        {PointSize[0.05],Point[pt[list,3]]},
        {PointSize[0.07],Point[pt[list,4]]}}]

ortho[list_]:=Graphics[{{AbsoluteThickness[4],
Line[{Take[list[[1]],2],Take[list[[2]],2],
Take[list[[3]],2],Take[list[[4]],2]}]},
{PointSize[0.05],GrayLevel[0.5],
 Point[Take[list[[1]],2]]},
          {PointSize[0.05],Point[Take[list[[2]],2]]},
          {PointSize[0.05],Point[Take[list[[3]],2]]},
          {PointSize[0.07],Point[Take[list[[4]],2]]}}]
```

```
rx[theta_]:={{1,0,0,0},
 {0,Cos[theta],Sin[theta],0},
 {0,-Sin[theta],Cos[theta],0},
 {0,0,0,1}}

ry[theta_]:={{Cos[theta],0,-Sin[theta],0},
 {0,1,0,0},
 {Sin[theta],0,Cos[theta],0},
 {0,0,0,1}}

rz[theta_]:={{Cos[theta],Sin[theta],0,0},
{-Sin[theta],Cos[theta],0,0},
{0,0,1,0},{0,0,0,1}}

tt[x_,y_]:={{1,0,0,0},
{0,1,0,0},
{0,0,1,0},
{x,y,0,1}}

sc[s_]:={{s,0,0,0},
 {0,s,0,0},
 {0,0,s,0},
 {0,0,0,1}}

SeedRandom[]

sv = 2*Random[];
rxv = Pi*Random[];
```

```
ryv = Pi*Random[];
rzv = Pi*Random[];
ttx = Random[];
tty = Random[];

trans={rx[Pi/6],ry[Pi/6],rz[Pi/6],
sc[sv].rx[rxv].ry[ryv].rz[rzv].tt[ttx,tty]};
figs=Table[four.trans[[i]],{i,4}];

basisx = {Transpose[figs[[1]]][[1]],
          Transpose[figs[[2]]][[1]],
          Transpose[figs[[3]]][[1]],
          {1,1,1,1}};
basisy = {Transpose[figs[[1]]][[2]],
          Transpose[figs[[2]]][[2]],
          Transpose[figs[[3]]][[2]],
          {1,1,1,1}};

unknx=Transpose[figs[[4]]][[1]];
unkny=Transpose[figs[[4]]][[2]];

cx=unknx.Inverse[basisx];
cy=unkny.Inverse[basisy];

ux=Inverse[Transpose[Inverse[basisx]].four].cx;
uy=Inverse[Transpose[Inverse[basisy]].four].cy;

a1[s_,xr_,yr_,zr_] := s*Cos[yr]*Cos[zr];
```

```
a2[s_,xr_,yr_,zr_] := s*(Cos[zr]*Sin[xr]*Sin[yr]-Cos[xr]*Sin[zr]);
a3[s_,xr_,yr_,zr_] := s*(Cos[xr]*Cos[zr]*Sin[yr]+Sin[xr]*Sin[zr]);

b1[s_,xr_,yr_,zr_] := s*Cos[yr]*Sin[zr];
b2[s_,xr_,yr_,zr_] := s*(Sin[xr]*Sin[yr]*Sin[zr]+Cos[xr]*Cos[zr]);
b3[s_,xr_,yr_,zr_] := s*(Cos[xr]*Sin[yr]*Sin[zr]-Cos[zr]*Sin[xr]);

SeedRandom[]
sol=FindRoot[{N[a2[s,xr,yr,zr]]==N[ux[[2]]],
  N[a3[s,xr,yr,zr]]
   ==N[ux[[3]]],
           N[b2[s,xr,yr,zr]]
            ==N[uy[[2]]],
           N[b3[s,xr,yr,zr]]==N[uy[[3]]]},
          {s,1+Random[]},
          {xr,Random[]*Pi},
          {yr,Random[]*Pi},
          {zr,Random[]*Pi},
          MaxIterations -> 30];
sol

cosol = {N[s /. sol],
rem[xr /. sol],
    rem[yr /. sol],
    rem[zr /. sol]}
N[{ux[[4]],uy[[4]]}]
N[{sv,rxv,ryv,rzv}]
N[{ttx,tty}]
```

```
N[{a1[s,xr,yr,zr] /. sol,
a2[s,xr,yr,zr] /. sol,
a3[s,xr,yr,zr] /. sol,
ttx}]

{a1[cosol[[1]],cosol[[2]],cosol[[3]],cosol[[4]]],
a2[cosol[[1]],cosol[[2]],cosol[[3]],cosol[[4]]],
a3[cosol[[1]],cosol[[2]],cosol[[3]],cosol[[4]]],
ttx}

N[ux]

N[{b1[s,xr,yr,zr] /. sol,
b2[s,xr,yr,zr] /. sol,
b3[s,xr,yr,zr] /. sol,
tty}]

{b1[cosol[[1]],cosol[[2]],cosol[[3]],cosol[[4]]],
b2[cosol[[1]],cosol[[2]],cosol[[3]],cosol[[4]]],
b3[cosol[[1]],cosol[[2]],cosol[[3]],cosol[[4]]],
tty}

N[uy]
```

Appendix C. Software Reference

The purpose of this appendix is to provide the reader with some insight into the construction (the word engineering does not necessarily apply) of the software system developed for this thesis. The execution of the three main components of the system, `writesat`, `matcher`, and `sens` is covered in the User's Guide, Appendix A.

Compilation and linking of the programs are handled through the use of a makefile. For clarity, the makefile is listed below, and each of the main source code files are discussed in turn.

C.1 Makefile

The following makefile is used for making all three of the programs on a Silicon Graphics workstation.

```
NRLIB = /usr/local/lib/libnewnr.a

matcher : matcher.o mysubs.o calcs.o rdfile.o
cc matcher.o mysubs.o calcs.o rdfile.o ${NRLIB} \
   primitives.o -lgl_s -lm -o matcher

matcher.o : matcher.c mysubs.h calcs.h matcher.h
cc -c matcher.c

mysubs.o : mysubs.c
cc -c mysubs.c

calcs.o : calcs.c
cc -c calcs.c

rdfile.o : rdfile.c drawsat.h primitives.h
cc -c rdfile.c

writesat : writesat.o primitives.o wprimitives.o readcontrol.o
cc writesat.o primitives.o wprimitives.o readcontrol.o \
   -lgl_s -lm -o writesat
```

```
writesat.o : writesat.c drawsat.h
cc -c writesat.c

primitives.o : primitives.c
cc -c primitives.c

wprimitives.o : wprimitives.c
cc -c wprimitives.c

readcontrol.o : readcontrol.c
cc -c readcontrol.c

sens : sens.o calcs.o calcs.h
cc sens.o calcs.o ${NRLIB} -lm -lgl_s -o sens

sens.o : sens.c
cc -c sens.c
```

C.2 Parameter Extraction: matcher.c

This program takes most of its execution options from the command line. Exercising the command line option sets global variables pertaining to main window size and scaling to the specified values. Otherwise the values retain the default values set at declaration.

The balance of execution options are read from the control file. The subroutine `readconfil`, contained in the file *mysubs.c* handles the reading of the control file and returns the input filenames in its parameter list.

It then opens the required windows. Widow operations are covered in the *Graphics Library Programming Guide* and the *Graphics Library Windowing and Font Library Programming Guide with Font Library Man Pages* publications.

Matrices for storing alignment points and basis images are set up using the *Numerical Recipes* utilities. Graphical objects are built using the `readvec` subroutine, contained in *mysubs.c.* `readvec` converts the 3D vector files into 2D vector images.

Next, the menuing system is initialized. Menuing on the SGI is covered in *Graphics Library Windowing and Font Library Programming Guide with Font Library Man Pages.*

The required devices are initialized to the event queue, then the event handling begins. Each of `matcher`'s functions is handled as a menu event by the **handlemenu** subroutine when the appropriate menu item is selected.

The calculation functions are handled by the **coeffs** and **parms** subroutines in the *calcs.c* file.

C.3 Supporting Subroutines: mysubs.c

This file contains miscellaneous routines used by **matcher**. It contains the routines **fatal**, **readconfil**, **readvec**, **drawcross**, **readref**, **drawplus**, and **drawbox**.

fatal is used in all programs. It exits the program and prints an error message to the C shell window when a fatal error occurs.

readconfil reads the control file input to **matcher**.

readvec reads the vector files created by **writesat** and converts them to 2D vector images. The conversion simply ignores the z coordinates in the vectors.

drawcross draws the crosshairs used to denote alignment points in the 2D images.

readref is reads the alignment points from the header of the a vector file.

drawplus draws the cross denoting the first corner of a zoom box.

drawbox draws the square denoting a the zoom box.

C.4 Numerical Routines: calcs.c

The routines for calculating the linear combination coefficients and extracting the parameters are contained in *calcs.c.*

The routine **coeffs** is called with the basis matrices and the alignment point positions in the sensor image, and returns the linear combination coefficients in the parameter list. It uses the Lower-Upper decomposition and backsubstitution routines from *Numerical Recipes in C.*

`seedrand` seeds the random number generator with the system time.

`parms` uses the *Numerical Recipes* LU decomposition and backsubstitution routines to find the composite transformation matrix, then calls the Newton-Raphson routine **`mnewt`** to find the alignment parameters.

`usrfun` contains the alpha and beta equations that describe the system to be solved. The alpha equations are the partial derivatives of (13) and (14), while the beta equations are (13) and (14) themselves.

`un2pi` reduces the rotation alignment parameters so that $0 < \theta \leq 2\pi$.

`test2` tests the equations from (13) and (14) that were not used in the Newton-Raphson extraction.

C.5 NSM Reader: rdfile.c

This file contains the routines that read the NSM format model (**`rdfile`**) and generate an SGI graphical object (**`makemodel`**).

`makemodel` generates the global graphical object called **`model`**. It calls **`rdfile`**.

`rdfile` reads the specified NSM file into a linked list of primitives. The beginning of the list is returned in the global variable called **`sat`**. Calling **`dprim`** begins the traversal of the list from the point specified as the argument (usually **`sat`**). **`dprim`** recurses until the entire list is rendered.

`dsat` draws the satellite by calling **`dprim`** and draws the spherical alignment point markers.

C.6 Image Generation: writesat.c

This file contains the main routine for the program **`writesat`**. The control flow of this program is very simple. During initialization, the execution options are read from the command line and the control file, the drawing window is initialized, the desired transformation is placed on the SGI's transformation stack, and the model file is read to a linked list as described above. The model is then displayed at the specified view. If a change in the transformation is desired, the new transformation is entered on the transformation stack. Once the desired view is achieved, the linked list is traversed, but instead of rendering lines to the screen, the endpoints of the lines are written to a file by the **`wsat`** and **`wprim`** routines. These routines are analogous to **`dsat`** and **`dprim`** described in the previous section.

C.7 Rendering: primitives.c *and* wprimitives.c

These files contain the the routines that render the NSM primitives to the screen (*primitives.c*) or to a file (*wprimitives.c*). Only the NSM primitives used in the DMSP used were implemented, and the arguments of each primitive are as defined in the NSM reference manual (2).

C.8 Performance Analysis: sens.c

This file contains the main routine for **`sens`**. The program repeatedly calls the parameter extraction routine, **`parms`**, calculates the error measures, and outputs its results to a file.

C.9 Summary

This appendix, taken with the comments in the code, will hopefully provide some insight into the organization of the software. The reader should bear in mind that the software exists only to show the utility of the parameter extraction method, and that this is not a software engineering thesis.

Bibliography

1. Brandt, Captain James R. *Real Image Visual Display System.* MS thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1992.
2. Center, Philips Laboratory Satellite Assessment. *SMT User's Guide* (Draft Edition). Kirtland AFB, NM: Philips Laboratory, 1992.
3. Foley, James D., et al. *Computer Graphics Principles and Practice* (Second Edition). New York: Addison-Wesley Publishing Company, 1990.
4. Huang, T. S. and C. H. Lee. "Motion and Structure from Orthographic Projections." *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 2., No. 5.* 536–540. 1991.
5. Jr., Captain Patrick Joseph Pond. *Three-Dimensional Medical Image Registration Using a Patient Space Correlation Technique.* MS thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1991.
6. Keller, Captain John G. *Identity Verification Through the Use of Face Recognition and Speaker Identification.* MS thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1993.
7. Kochan, Stephen G. *Programming in C.* Indianapolis: Hayden Books, 1983.
8. McLendon, Patricia. *Graphics Library Programming Guide.* Mountain View, CA: Silicon Graphics, Inc., 1991.
9. Pond, Captain David L. *A Synthetic Environment for Satellite Modelling and Satellite Orbital Motion.* MS thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1992.
10. Press, William H., et al. *Numerical Recepies in C: The Art of Scientific Computing.* New York: Cambridge University Press, 1988.
11. Reimann, Robert, et al. *Graphics Library Windowing and Font Library Programming Guide with Font Library Man Pages.* Mountain View, CA: Silicon Graphics, Inc., 1991.
12. Thau, Robert S. "Application of Generalized Radial Basis Functions to the Problem of Object Recognition." *SPIE Vol. 1469 Applications of Artificial Neural Networks II.* 37–47. SPIE, 1991.
13. Ullman, Shimon and Ronen Basri. "Recognition by Linear Combination of Models," *MIT AI Memo No. 1152* (August 1989).
14. Ullman, Shimon and Ronen Basri. "Recognition by Linear Combinations of Models." *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 992–1006. October 1991.
15. Winston, Patrick Henry. *Artificial Intelligence* (Third Edition). New York: Addison Wesley, 1992.

16. Wolfram, Stephen. *Mathematica: A System for Doing Mathematics by Computer* (second Edition). New York: Addison-Wesley Publishing Company, 1991.
17. Zahirniak, Captain Daniel R. *Characterization of Radar Signals Using Neural Networks*. MS thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1990.

Vita

Captain Martin S. Chin was born on 23 September, 1967 in Kingston, Jamaica. He graduated from Mandeville High School in Mandeville, Louisiana in 1984. He then attended Tulane University in New Orleans, Louisiana where he was awarded the degree of Bachelor of Science in Electrical Engineering in May of 1988. After his commissioning through AFROTC, he worked as an electrical engineer on an airborne bathymetry sensor for the Navy Ocean Research and Development Activity at the Stennis Space Center, Mississippi. In March of 1989, he began his first assignment on active duty for the Air Force as an Advanced ICBM Systems Concepts Project Officer at the Ballistic Missile Organization at Norton AFB, California. He received a Master of Science in Systems Management from the University of Southern California in August of 1991. In May of 1992, he entered the School of Engineering a the Air Force Institute of Technology.

Permanent address: 22452 Swordfish Drive N.
Boca Raton, Florida 33428

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
December 1993

3. REPORT TYPE AND DATES COVERED
Master's Thesis

4. TITLE AND SUBTITLE
Satellite Attitude Determination Using Linear Combination of Models

5. FUNDING NUMBERS

6. AUTHOR(S)
Martin S. Chin, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GE/ENG/93D-05

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Capt G. Tarr
PL/LIMI
Kirtland AFB NM 87117

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

In satellite analysis, it is often necessary to determine the attitude of a satellite from a two dimensional image of the satellite. This thesis describes work performed to determine the attitude of a satellite relative to a reference position. The attitude is described in terms of the scaling, rotation and translation of the reference that will result in a pose that can be projected to form the image of interest. The techniques used are based on finding coefficients for the linear combination of basis images and decomposition of a composite transformation matrix. The methods were initially modeled in *Mathematica* and applied to simple objects, then implemented on a Silicon Graphics Workstation and applied to a model of an actual satellite. An accurate estimate of the pose of a satellite in a 2D image can be estimated based on the information contained in the image and a 3D model of the satellite or a set of images of the satellite at known orientations. Results of a performance analysis of the SGI implementation, a user's guide, source code in *Mathematica* and a software reference are included.

14. SUBJECT TERMS
attitude determination, linear combination, basis image, linear transformation, satellite modeling

15. NUMBER OF PAGES
95

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
UNCLASSIFIED

18. SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

19. SECURITY CLASSIFICATION OF ABSTRACT
UNCLASSIFIED

20. LIMITATION OF ABSTRACT
UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std Z39-18
298-102